Dokument Typ:
Document Type: Manual

Titel:
Title:

CGS V4.4.0 User Manual

Lieferbedingungs-Nr.: DRL/DRD No.:	N/A	**Klassifikations Nr.:** Classification No.:	N/A
Produktgruppe: Product Group:	COL–RIBRE–MA–0028	**Konfigurationsteil-Nr.:** Configuration Item No.:	1214 597
Schlagwörter: Headings:	CGS User Manual	**Produktklassifizierungs-Nr.:** Classifying Product Code:	8–QA
		Freigabe Ordnungs-Nr.: Release Order No.:	N/A

Bearbeitet: Prepared by:	CGS – Team	**Abteilung:** Department:	RIO6	**Firma:** Company:	DASA RI
Geprüft: Agreed by:		**Abteilung:** Department:		**Firma:** Company:	
Genehmigt: Approved by:		**Abteilung:** Department:		**Firma:** Company:	
Genehmigt: Approved by:	R. Zimmermann	**Abteilung:** Department:	RIO62	**Firma:** Company:	DASA RI

DCR-Daten/Dokument-Änderungsnachweis /Data/Document Change Record

Überarbeitung Revision	**Datum** Date	**Betroffener Abschnitt/Paragraph/Seite** Affected Section/Paragraph/Page	**Änderungsgrund/Kurze Änderungsbeschreibung** Reason for Change/Brief Description of Change
1/–	03.11.1995	All	First issue
2/–	29.02.1996	All	Compatible with CGS Build 3.0
2/A	30.04.1996	2.2, 6.2.3, 6.2.6, 7.1, 7.2, 7.3, Appendix C, Appendix F	Compatible with CGS V3.1.0
2/B	10.09.1996	1, 2, 3.1.2, 7.1.12, 7.2, 8.1, 8.1.9, 8.2.3, 9, Appendix C, Appendix D, Appendix F	Compatible with CGS V3.1.2
2/C	14.02.1997	All	New screenshots
		2, 3.1.1, 3.1.2, 4.3, 6.2.3, 7.1.5, 7.1.11, 7.3, 8.1.2, 9, Appendix C, Appendix D	Compatible with CGS V3.1.3
		10	New chapter (added)
		Appendix I	New appendix (added)
2/D		NO TAG	Discrepancy Notice *DMS–R BM00091*
		All	Compatible with CGS V4.1.0.7 (see change bars to identify changes. Sorry: No change bars have been provided in ”Model Development” and ”Model Observation and Control”)
2/E		All	Compatible with CGS V4.1.1
2/F	20.5.98	Appendix J	Correct CGS system limitations
		chapter 8.1.	Add hints how to deal with disc space/table spaces
		chapter 11	Add CGS Administration chapter
		Appendix K.	New chapter to list all user definable configuration file parameters
2/G	2.10.98	Ch.2	Update versions of referenced Documents
		8.2.2.3.1, Appendix K	Added description of HLCL login sequence.
		9.3.3 / 9.3.4	Describe buttons for FA_SAS startup/display in TEV
		Appendix H	GET_ENV. and START_HCI_APPLICATION
		Appendix I	New Version of UCL System Libraries
		Appendix J	Update Limitations
		Appendix F	Update CGS API
		Appendix L	Add new appendix: Screen Setup

DCR-Daten/Dokument-Änderungsnachweis /Data/Document Change Record

Überarbeitung Revision	**Datum** Date	**Betroffener Abschnitt/Paragraph/Seite** Affected Section/Paragraph/Page	**Änderungsgrund/Kurze Änderungsbeschreibung** Reason for Change/Brief Description of Change
1/–	03.11.1995	All	First issue
2/–	29.02.1996	All	Compatible with CGS Build 3.0
2/A	30.04.1996	2.2, 6.2.3, 6.2.6, 7.1, 7.2, 7.3, Appendix C, Appendix F	Compatible with CGS V3.1.0
2/B	10.09.1996	1, 2, 3.1.2, 7.1.12, 7.2, 8.1, 8.1.9, 8.2.3, 9, Appendix C, Appendix D, Appendix F	Compatible with CGS V3.1.2
2/C	14.02.1997	All	New screenshots
		2, 3.1.1, 3.1.2, 4.3, 6.2.3, 7.1.5, 7.1.11, 7.3, 8.1.2, 9, Appendix C, Appendix D	Compatible with CGS V3.1.3
		10	New chapter (added)
		Appendix I	New appendix (added)
2/D		All	Compatible with CGS V4.1.0.7
			Discrepancy Notice *DMS–R BM00091*
2/E		All	Compatible with CGS V4.1.1
2/F	20.5.98	Appendix J	Correct CGS system limitations
		chapter 8.1.	Add hints how to deal with disc space/table spaces
		chapter 11	Add CGS Administration chapter
		Appendix K.	New chapter to list all user definable configuration file parameters
2/G	2.10.98	Ch.11	Add description for startup/shutdown of tools / system
		ch. 8	Add description of HLCL Login Sequence
		App. D	Update error messages
		App. F	Remove outdated parts
		App. G	Recreate Index
		App. I	Latest Versions of UCL System Libraries
		App. J	Add some limitations
		Add App. L	Screen Setup Defnitions

DCR-Daten/Dokument-Änderungsnachweis /Data/Document Change Record

Überarbeitung Revision	Datum Date	Betroffener Abschnitt/Paragraph/Seite Affected Section/Paragraph/Page	Änderungsgrund/Kurze Änderungsbeschreibung Reason for Change/Brief Description of Change
2/H	31.3.99	8.2.2.3.2.6	SAS Status
		9.3.2, 9.4.3, 9.4.4	TEV : Execution Sessions selection, Raw Data Dump tool : choose ADUs and GDUs from Archive files,Data Set tool : possibility to choose ADUs related to a measurement
		4.3.8	Message Handler: regular expressions for test node/group name.
		4.2.3	Updated task selector figures.
		8.2.2.3.2.9	Revised Synoptic Displays section.
		8.2.2.5	Replaced Online Test Control configuration file by 4.2 version
		7.6	New chapter to describe CGS standard enditem types
		App. J	Update some limitations
		App. K	Update some config parameter
		ch. 8.1.2.7 / ch 8.1.2.8 / ch. 11	Restructure chapters on System Administration and TRDB Administration / Resource Considerations: Integrate in ch.11
2/I	16.7.1999	2	Update Document Versions
		3,11	Add Description of SUN as test node
		5.1	Add SWES Invokation
		7.2	Update SAS Implementation Procedures
		7.6	Add restrictions to definition of statecodes in GDUs Describe constraints on monitoring definitions Update acc. to implemented extensions in CGS
		8.2.2.9.6	Replay Session Control: Use of MTP and other constraints
		Figure 8–10	Updated TSCV Replay Properties window
		8.3.2.3.2.8	Updated Test Node Status window (new HK items)
		11.1.1	Inserted section: Turn On/Boot CGS Hardware
		9.4.3	TEV : Raw Data Dump tool. Describe the interruption button and the warning message about the size of the archive files.
		9.4.4.2	TEV : Data Set tool. Describe the interruption button. Describe the **Choose ADUs from Archive files...** and **Choose ADUs from MDB...** buttons.
		9.2	TEV : Using TEV in Batch mode Additional information inserted.
		ch. 10.3.1.1	Extend Recovery scripts with menu 1.7: Transfer lost files
		App. I: UCL System Libs	Updated Libraries acc. to COL–RIBRE–IRN–8003 Iss 1/C
		App. M	New Appendix to describe SAS_FILE_IO

DCR-Daten/Dokument-Änderungsnachweis /Data/Document Change Record

Überarbeitung Revision	Datum Date	Betroffener Abschnitt/Paragraph/Seite Affected Section/Paragraph/Page	Änderungsgrund/Kurze Änderungsbeschreibung Reason for Change/Brief Description of Change
2/K	31.3.2000	2	Include CGS ICD as referenced document
		6.3	Describe new functionality of CLS Editor for Derived Values;
		7.1.2.2	Update restrictions defined for a test evaluator user
		7.1.12	New HK Values acc. to COL–RIBRE–CGS–IRN–8057 Issue 1/B
		7.6	Make some attributes optional; delete some attributes acc. to COL–RIBRE–CGS–IRN–8066 Issue 1/B
		7.6, 8.3.3	New chapters to describe derived values Add Conditions and TC Verification (see COL–RIBRE–CGS–IRN–8072, –8073 and 8074)
		7.6.3.8	New Data Source Attribute acc. to COL–RIBRE–CGS–IRN–8063 Issue 1/A
		7.7	New chapter to describe consistency checking
		11.7.2.3	Revised cleanup (disks) section.
		8.2.2.11.3.10	Description of new window (SPR–6470)
		8.2.2.11.3.24	hint for closing session
		8.3	Updated window snapshots. Major updates in Graph Facility (new MDB Browser), Explorer (new HCI application), Synoptic Displays (acquisition / monitoring / processing status).
		9.3.1	TEV: Evaluation Session Definition : new button :"Session Size"
		9.3.2.1	TEV: Select Execution Sessions : new button :"Session Size"
		9.4.1	TEV: Data Evaluation Tools: Generalities : – Definition includes User Events to define a time frame and the information "Initial Time Frame" – Print result operations use postscript
		9.5.3	TEV: Data Set/ Events Merger tool
		ch. 10.3.1.1	Extend Recovery scripts with alternative 1.8: Close aborted session to 'Normally Closed'
		11.3	Insert new chapter to describe installation of CGS External Software Items or CGS Quick Patches
		D–3.2.2.4	New error message inserted (SPR–6961)
		D–3.3.2	Updated message type mapping acc. to COL–RIBRE–CGS–IRN–8082.
		App. I	Update UCL Ground_Library acc. to COL–RIBRE–CGS–IRN–8076
		App. N	New Appendix to describe MDB consistency checks

DCR-Daten/Dokument-Änderungsnachweis /Data/Document Change Record

Überarbeitung Revision	**Datum** Date	**Betroffener Abschnitt/Paragraph/Seite** Affected Section/Paragraph/Page	**Änderungsgrund/Kurze Änderungsbeschreibung** Reason for Change/Brief Description of Change
2/L	31.10.2000		Updates acc. to CGS_V4.4.0
		p 7–33	Correction: HK Values 321–324
		7.1.12	new HK value 1013
		7.6	Updates acc. to MDB structure changes defined in COL–RIBRE–IRN–8107 Issue 1/A
		8.3.2.3.2.3	New Measurement/Monitoring Window
		8.3.2.3.2.4	New GO/NOGO Window
		8.3.2.3.2.11	Input Dialog (new layout, TC authorization)
		8.3.2.3.2.10.5	Synoptic Display Control Panel (new)
		8.3.3.3	explain new concept of command authorization
		8.3.3.11	new configuration parameter for TES
		Appendix D.3.3.2	new TES error messages
		Appendix F	add ACKNOWLEDGE_COMMAND procedure
		Appendix H	New version of Start_HCI_Application command.
		Appendix I	Update UCL Ground_Library acc. to COL–RIBRE–IRN–8105 Issue 1/A
			Update UCL Ground_to_OB_Lib acc. to COL–RIBRE–IRN–8101 Issue 1/A
		Appendix K	Update Config Parameter
		Appendix M	new operation for FILE_IO_LIB: ADD_FILE_TO_TEST_SESSION

CONTENTS

FIGURES

1 INTRODUCTION

1.1 Identification and Scope

This document is the CGS V4 User Manual.

1.2 Purpose

This Manual provides the CGS user with a top–level introduction how to use CGS for software development, mission preparation and system test purposes.

1.3 How this document is organised

This Manual has been organised to provide the user the following:

– an overview of the concept behind the major CGS functions
– an introduction how to use the major CGS functions

For detailed information on the use of some CGS tools the user will have to refer to the lower level CGS component Reference Manuals and User Manuals identified in Chapter 2 'Documents' of this Manual. This is particularly the case with the Software Development Environment (SDE) and the Mission Database Application (MDA) and with the Window Definition Utilities (FWDU and GWDU). In these cases this Manual provides the user with information regarding the top level CGS application of such tools.

Chapter 1	**Introduction** *Provides a general overview of the Manual*
Chapter 2	**Documents** *Lists all referenced documentation*
Chapter 3	**CGS – The Development and Test Support System** *Describes the CGS functions, hardware/software environment, CGS concept and available CGS configurations for each user task*
Chapter 4	**CGS General Setup Tasks** *Describes how to setup the CGS user environment, how to start CGS and an overview of the error message handling concept*
Chapter 5	**Development Support Services** *Describes how to use the CGS ADA/C programming programming environments and how to use the documentation preparation tool*
Chapter 6	**Mission Preparation Tasks** *Describes how to build Mission Configurations with CGS and how to prepare the contents of the data forming part of a mission configuration*
Chapter 7	**Test Preparation** *Describes how to setup a test configuration, develop APs, SASs, ground synoptic displays and simulation models*

Chapter 8	**Integration and Test** *Describes how to setup the test environment with CGS and how to execute a test session*
Chapter 9	**Test Evaluation** *Describes how to use CGS to evaluate and report on the results of a test execution session*
Chapter 10	**TRDB Tools** *Describes how to use CGS to maintain the Test Result Data base (TRDB)*
Appendices	*Provides detailed information with respect to end items, error messages, configuration parameter, SAS API and test–related HLCL commands*

2 DOCUMENTS

2.1 CGS User Manuals

The following manuals are part of the CGS User Manual delivery and build together with this manual the complete CGS User Manual documentation set.

2.1.1 MDA Manuals

2.1.1.1 COL–RIBRE–MA–0029–00
MDA Users and Operations Manual
Issue 4/B, 31.03.2000
(covers documents 2.1.1.2, 2.1.1.3 and 2.1.1.4)

2.1.1.2 COL–RIBRE–MA–0030–00
MDA Introduction Manual
Issue 3/B, 04.04.1997

2.1.1.3 COL–RIBRE–MA–0031–00
MDA Reference Manual
Issue 4/C, 31.03.2000

2.1.1.4 COL–RIBRE–MA–0018–00
MDA Administration Manual
Issue 4/B, 31.03.2000

2.1.1.5 COL–RIBRE–MA–0035–00
DADI–MA Users and Operations Manual
Issue 4/–, 01.09.1997
(covers documents 2.1.1.6, 2.1.1.7 and 2.1.1.8)

2.1.1.6 COL–RIBRE–MA–0037–00
DADI–MA Introduction Manual
Issue 3/–, 04.04.1997

2.1.1.7 COL–RIBRE–MA–0032–00
DADI–MA Reference Manual
Issue 4/–, 01.09.1997

2.1.1.8 COL–RIBRE–MA–0036–00
DADI–MA Administration Manual
Issue 4/–, 01.09.1997

2.1.1.9 COL–RIBRE–MA–0046–00
SID Range Tool Users and Operations Manual
Issue 1/–, 15.09.1997

2.1.2 WDU Manuals

2.1.2.1 UM–114–001–ROV
GWDU User’s Manual and Operations Manual
Issue 1.4, 24.02.1999

2.1.2.2 Deleted

2.1.2.3 UM/114/002/ROV
FWDU User Manual
Issue 2.2, 15.08.1999

2.2 Referenced User Manuals

2.2.1 SDE User Manuals

2.2.1.1 SDE – User Manual Set

2.2.1.1.1 *No Number*
Life*CYCLE User Documentation –
Life*CYCLE User Manual
Issue 2.1, May 95

2.2.1.1.2 *No Number*
Life*CYCLE User Documentation –
Life*LINK User Manual
Issue 2.0, May 95

2.2.1.1.3 *No Number*
Life*CYCLE User Documentation –
Configuration Management System User Manual
Issue 2.0, May 95

2.2.1.1.4 *No Number*
Life*CYCLE User Documentation –
Generic Tool Integration User Manual
Issue 2.0, May 95

2.2.1.1.5 *No Number*
Life*CYCLE User Documentation –
Life*ADA User Manual
Issue 1.1, Jul 95

2.2.1.1.6 *No Number*
Life*CYCLE User Documentation –
Life*ADA Integration Manual
Issue 1.1, Jul 95

2.2.1.1.7 *No Number*
Life*CYCLE User Documentation –
Interleaf Integration User Manual
Issue 2.0, May 95

2.2.1.1.8 *No Number*
Life*CYCLE User Documentation –
Logicscope Life*ADA Integration User Manual
Issue 1.0, Aug. 95

2.2.1.1.9 *No Number*
Life*CYCLE User Documentation –
Life*CODE Users' Manual
Issue 0.7 (DRAFT), Aug. 94

2.2.1.2 SDE – Technical Notes

2.2.1.2.1 *No Number*
Life*CYCLE Technical Note –
Tracing in Interleaf Documents
Issue 1.0, Aug. 95

2.2.1.2.2 *No Number*
Life*CYCLE Technical Note –
Tracing in Interleaf Documents
Issue 1/A, Aug. 95

2.2.1.2.3 NFJ/TN/95/001
Reporting Interface for Life*CYCLE
Issued: 06.05.1995

2.2.1.2.4 CSD/PRJ/COL/SPEC/20
A description of how Life*CYCLE
Configuration Management System (CMS)
automates key Columbus CM activities
Issue 1.0, 03.08.1995

2.2.1.2.5 CSD/PRJ/COL/SPEC/19
Life*CYCLE Support for Development of SW Entities
Issue 1.0, 21.07.1995

2.2.2 VICOS Manuals

2.2.2.1 COL–SBI–600–SUM–002
DBS – FA SAS User Manual
Issue 2, 09.01.1996

2.3 Other Referenced Documents

2.3.1 COL–RIBRE–MA–0025
CGS Installation Manual
Issue 4/F, 31.03.2000

2.3.2 COL–RIBRE–ICD–0069
MDB Standard Entities and Application Programming Interface
Issue 4/A, 31.03.1999

2.3.3 COL–RIBRE–STD–0010
User Control Language (UCL) Reference Manual
Issue 4/–, 13.03.1998

2.3.4 COL–RIBRE–STD–0009
High Level Command Language (HLCL) Reference Manual
Issue 2/A, 13.03.1998

2.3.5 COL–RIBRE–STD–0008
Reference Manual for Crew Procedure Language and Software Commanding
Issue 1/E, 18.05.1998

2.3.6 STD 1216804
Ground Human–Computer Interaction Standards
Issue 5/B, 28.02.1993

2.3.7 Sun Ada Programmer’s Guide, SUN Microsystems
March 1992

2.3.8 ISBN 0–201–52364–7
Open Look Graphical User Interface. Application Style Guidelines

2.3.9 ISBN 0–201–52365–5
Open Look Graphical Interface. Functional Specification

2.3.10 COL–RIBRE–ICD–0025
CGS Interface Control Document
Issue 4/A, 31.03.1999

3 CGS – THE DEVELOPMENT AND TEST SUPPORT SYSTEM

3.1 Overview of CGS

3.1.1 Function and Purpose

The COLUMBUS Ground System (CGS) constitutes the set of products which support or enable various activities performed during Design and Development and the Integration, Test and Qualification of the Flight Configurations. CGS will be utilized in different ground based Facilities of various Space Programmes. Major Facilities currently identified are:

- Software Design and Development Facility (SDDF),
- Electrical Ground Support Environment (EGSE),
- Software Integration and Test Environment (SITE).

CGS will support the following functionality:

Design and Development Support for Ground and Flight Systems including:

- Columbus Ground System Infrastructure (CGSI) provides the basic service layer to all CGS S/W Applications, supporting Servers (SUN–Ultra), Test Nodes (HP 9000 Series or SUN Ultra) and graphical Workstations (SUN–Sparc/Ultra) under the UNIX Operating System (Solaris or HP_UX).
  It provides a Top Level User Interface with menus to startup remaining applications and a message window to present messages from the various tools to the user.
- Software Development Environment (SDE) provides standard S/W Life Cycle support:
  - S/W Requirements Phase Support including Definition and Analysis Support in SADT for S/W Requirements and Definition Support for S/W Interface Control Documents,
  - S/W Architectural / Detailed Design Phase Support including HOOD Methodology for Application S/W Architectural and Detailed Design and Program Design Language for Application S/W Detailed Design,
  - S/W Coding / Testing Support including Syntax Editing for Ada and C source code. S/W Compiler Systems and additional S/W Development Tools might be integrated to through a standard generic Compiler and Tool Interface,
  - Traceability Support over the S/W Life Cycle including traceability between different contractual level (”vertical traceability”) and traceability from requirements to design on the same level (”horizontal traceability”)
  - S/W Documentation Handling,
  - and S/W Configuration Management, based on contractual Configuration Items.
- The *Software Entity Software (SWES)* provides support for Ada Precompilation of SW and Transfer of Onboard SW into the Mission Database (MDB).
- The *Mission Database Application (MDA)* constitutes the set of utilities which support or enable various activities typically performed during the preparation phase of a checkout /

simulation test or mission and provides data entry and reporting, configuration management and support to off-line generation of onboard database/flight image. MDA is centered around the Mission Database which serves as a central repository for all test / mission–related information.

- The *Columbus Language System (CLS)* comprises several language related software components for UCL and HLCL:
  - The CLS Editor is the XView based user front–end for editing UCL or HLCL command sequences as well as for the generation of CPL sequences and parameter list editing and for the specification of SW Commands within the interactive database environment of MDA,
  - The UCL Compiler translates automated procedures (APs) and libraries written in the User Control Language (UCL) into an intermediate code (I–Code), which can be executed by the Test Execution Software (TES),
- The *Ground Window Definition Utility (GWDU)* provides functionality to generate checkout and simulation orientated ground synoptic displays. These layouts contain animated functional drawings which will be used to display e.g. checkout, simulation, subsystem and payload status information and to read in dedicated commands from the human user. This product will be based on the Ground Symbol and Display Standard.
- The *Flight Window Definition Utility (FWDU)* provides functionality to generate flight synoptic displays. These layouts contain animated functional drawings which will be used to display subsystem and payload status information and to read in dedicated commands from the human user. This product will be based on the Flight Symbol and Display Standard.
- ***Integration, Test and Qualification Support*** for Flight Systems:
  - The *Test Setup, Configuration and Verification Software (TSCV)* configures a required system configuration for a Checkout Test / Simulation. It enables the test nodes to be activated for a given test. TSCV implements the generation of a test session in the master archive. TSCV further supports the user to identify and control the currently used S/W Configuration.
  - The *Human Computer Interface (HCI)* provides all services related to user input / output on workstations during execution of a Checkout Test / Simulation Session. It provides different interfaces to the window system and the HCI devices (keyboard,mouse,screen) and includes services for synoptic display update, user help and user guidance through test operations. It provides the commanding interface and the message output during online test execution.

    HCI includes the HLCL Command Window, which constitutes the interactive commanding interface of different applications running on various workstations in the ground system. It interprets and executes interactive commands and automated command sequences written in the High Level Command Language (HLCL),
  - The *Test Execution Software (TES)* implements the support for test operations and automatic testing / monitoring of the Unit under Test (UUT). It is driven by the test definitions

done by MDA and the configuration setup by TSCV. It provides a generic data and control interface to the UUT and all services required for realtime enditem data processing.

- *Data Base Services (DBS)* provides low level management and access to checkout test related result data items stored in the Test Results Database or files.
- *Test Evaluation Software (TEV)* provides all services to evaluate a.m. data generated and stored during checkout test execution. It provides services for data selection and data presentation. It further implements the final archiving of test results as well as the selective exporting / importing of parts of the Test Result Database.
- *Network Software (NWSW)* provides low level message based interprocess communication as well as file transfer and directory services in a non NFS environment.
- *Timing Services Software (TSS)* provides the synchronization of local computer clocks in a distributed environment with respect to the actual (local) time. In addition, TSS also provide low level SMT distribution, access and handling.
- The *Core Simulation Software (CSS)* provides support for Simulation Model Development, Simulation Preparation and Execution / Control of Models.

3.1.2 CGS Hardware and Commercial Software Environment

It should be noted that CGS is a 'pure' software system. The user has therefore to separately procure the necessary hardware and commercial software environment.

CGS operates in the following hardware environment:
Note: for an more detailed definition refer to the custumer support services.

Server:	SUN SPARC, memory ≥ 64 MB, disk ≥ 2 GB
Workstation:	SUN SPARC ≥ 64 MB, color screen
Printserver config.:	SUN SPARC, 16 MB memory

In addition the following hardware is required for specific tasks:

Test node configuration:	HP9000/7xx, memory ≥ 128 MB, disk ≥ 2 GB
Test node configuration:	SUN Ultra, memory ≥ 128 MB, disk ≥ 2 GB
Simulation node:	teraforce–10 or sun, memory ≥ 128 MB, disk ≥ 2 GB

The required commercial software environment to run CGS is as follows:

For the SUN:	SOLARIS 2.6 Oracle 7.3.4 Dataviews 9.7a/Run–Time Visual Works 3.0 (only for simulation tool) Sammi 3.0.12
For the HP:	HP–UX 10.20.A Alsys Compiler 5.5.4 + Alsys patch CG2V553–U2 (for SAS development – see Section 7.2)
For the FORCE:	SOLARIS 2.5 Visual Works 3.0 FORCE VME Bus Driver 1.14

3.2 The CGS checkout and test system

The purpose of this section is to provide the user with an introduction to the concept of the CGS checkout and test system.

3.2.1 General

Operationally CGS functionality is logically partitioned into two systems:
The **off–line system** supporting the preparation and management of software, hardware and data.
The **on–line system** supporting the integration, checkout and qualification testing of hardware and software.

There are a number of Design Goals for CGS software, involved in the checkout and test operations.

Vertical Commonality

- Where possible, the same test system is used for different test levels eg from Assembly level to System level.
- A test object oriented user interface allows for execution of tests independently of the actual test system configuration, eg an Automated Procedure shall be executed (without change) on different hardware or system configurations.
- Transparent access to the standard services of the system is provided with the use of logical names (”pathnames”) to refer to enditems.

Open Architecture

- Additional Hardware and Software Units can be integrated into a test system, thus meeting different requirements coming from a variety of Users and test objects (UUT: Unit Under Test).

Integrated Toolset

- Integrated tools enable Definition, Execution/Monitoring and Evaluation of a Test.

3.2.2 Basics

To introduce the user into the terminology as well as to give an overview on the CGS products used for testing, a short description is given in the following section.

The checkout and test system based on CGS products can be used for

- test preparation
- test configuration
- test execution
- test evaluation

Test has to be understood as a general term standing for

- integration
- troubleshooting
- checkout

- and verification activities

Note that the checkout and test system is sometimes called VICOS (Verification, Integration and Checkout Software) although infact this is a generic name covering a number of tools.

The **Item Under Test** is the hardware or software being tested (e.g. Ground APs and UUTs) . This may be a ground or flight component.

The **Test Harness** is the framework of software, hardware and data especially produced for testing the Item under Test. It consists of Test Controllers, Test Stubs and Test Data that are needed for the test.

The **Test Controllers** are the software and hardware that control execution of a test. Software test control can range from Ada Tool support to CGS Products: HCI/TES/TSCV. Hardware Test Control can range from a variable resistor to the EGSE Hardware Control System.

Test Stubs are the software and hardware that provide dummy functionality for components not yet available (software 'stubs' and hardware 'dummies'). Software stubs can range from empty software modules through complex Simulation Models to a qualified flight software subsystem. Hardware dummies can range from a short circuit through an EGSE SCOE system to qualified flight hardware subsystem.

CGS Add Ons are required to provide Compilation environments and SW Analysis software, provide interfaces to the Test harness and Test Equipment and also possibly specialist hardware to support development.

System Hardware provides the platform on which both the On–line and Off–line environments execute. For product level production these are usually the same hardware with specialist hardware labelled CGS hardware add–ons. For system testing these platforms are significantly different.

The CGS Software products forming the checkout and test system are:

- MDA – the Mission Database Application
- TSCV – the Test System Configuration and Verification Software
- TES – the Test Execution Software
- TEV – the Test Evaluation Software
- DBS – the DataBase Services
- HCI – the Human Computer Interface
- PLATFORM Services:
  CGS Infrastructure (CGSI)
  Network Software (NWSW)
  Time Synchronisation Services (TSS)

Test preparation, as described in the following sections, is mainly the description of the test article (unit under test) as well as the test system (test equipment) in terms of re–usable configuration database enditems.

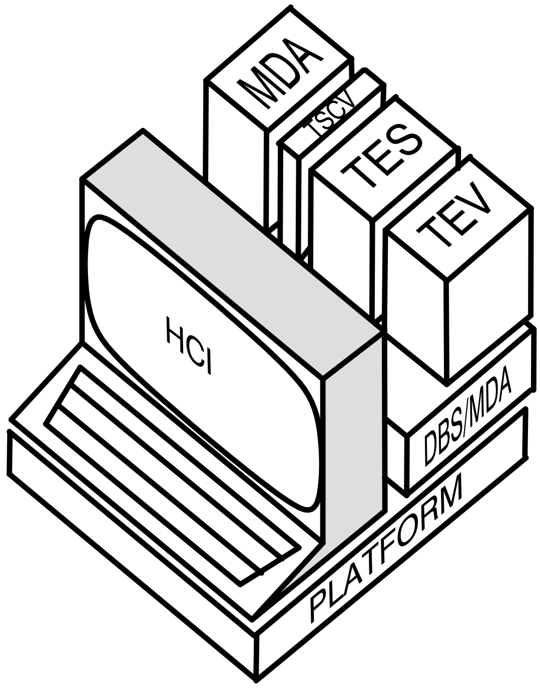


Figure 3–1 : *Symbolic representation of CGS products building the checkout and test system*

Tailoring the checkout and test system to a specific test system configuration

The architecture of the unit under test and the test equipment should determine the structure of the name tree.

Re–use of onboard configuration data by proper mapping can be achieved.

Note that the User Profiles which
– describe the access right of users to system resources and
– describe the system environment for the user
are no longer part of the MDB.

Before the software products listed earlier can be activated and loaded, the following definitions must have been done:

- Definition of UUT end–items and UUT H/W – S/W configurations within the Mission DB (MDB)
- Definition of simulation models replacing the UUT for special testing purposes
  and additionally the measurements and stimuli which connect the model
  with the system
- Definition of EGSE acquisition and stimuli end–items within the MDB
  This includes for the measurement descriptions:
  – Describe the attributes of a measurement point
  (in the test equipment or the unit under test)
  – Physical acquisition information for hardware devices
  – Calibration from raw to engineering values
  – Monitoring
  – Emergency actions in case of limit violations

  and for the stimuli descriptions:

  – Describe a stimulus inside the test equipment or to the unit under test
  – Configuration information for the hardware device
  – Parameter definition
  – Related actions in the test system

- Definition of nodes used in the test configuration
  – Describe the role of (computer) nodes
  – Make nodes accessible via nametree
  – Perform logical to physical name translation

- Definition of the checkout and test system S/W configuration to be loaded to each node (this includes the definition of the test configuration tables specifying the allocation)
  – Describe the version and location of EGSE Software

- Development of Specific Application S/W (additional Ada modules on top of the test system (only needed for a specific test))

- Definition of general UCL and HLCL sequences for EGSE activation/ shutdown/ operational control

Preparing a specific test
A specific test is be defined in the following way:

- Establishment of a test plan providing information on test requirements, test environments and configuration control of the EGSE and UUT. This will be done with support from the CGS off–line system or manually.

- Development of operational test definitions (shall be performed in the MDB) which includes:

  - Definition of UCL/HLCL sequences
    The user may either edit and compile automatic procedures (AP) written in UCL or define interactive command sequences (HLCL sequence) that allow symbolic operations and sequencing of single commands.

  - Definition of windows/pictures
    The user may define colored graphical representations (pictures) of the UUT and the EGSE and of the actual status of the devices during online operations. The picture can be established by an interactive drawing tool (WDU) that defines graphical symbols of high resolution and their references to end–items defined in the DB.

  - Definition of automatic monitoring
    The user may define automatic monitoring of end–items assigning multiple limit sets to end–items and defining automatic actions to be performed in a (dangerous) out of limit situation. Such actions may be UCL procedures, telecommands or commands to the EGSE H/W (stimuli).

  - Definition which predefined configuration shall be loaded for test
    To enable the test nodes for fast execution of UCL procedures, data formatting etc. the user has to allocate a specific function (e.g. monitoring of a specific UUT sub–system) to a test node. This results later in the load of specific tables (e.g. all tables of a subtree which define the configuration of the resp. SS) from the DB to the test nodes memory.

Furthermore the user defines the default test system configuration be specifying those nodes that shall take part of the test.

3.2.3 Key feature: the Database driven system

Figure 3–2 shows the way operations are driven by the different databases:

Operations are driven by a configuration database which

- contains description of the UUT and the test system.
- contains also the automated test procedures, synoptic picture definitions,etc
- utilizes SDDF version control and user authorisation mechanisms
- data are logically grouped

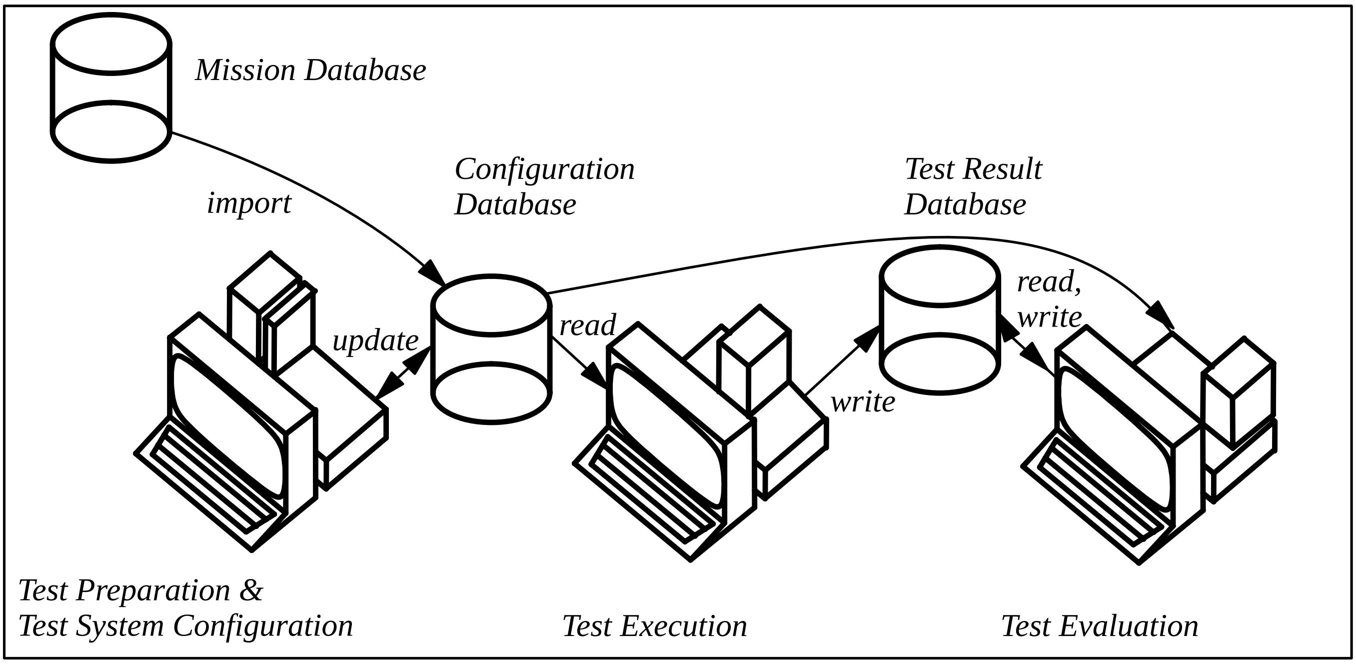


Figure 3–2 : *Anticipated flow of checkout operations*

Test results are stored in the test result database which

- contains test result data
- serves as intermediate storage of test results on magnetic media
- serves as final archive = long term storage of data on other media (currently optical disc)
- serves as Master archive covers both, intermediate and long term storage

3.2.4 Key feature: the Distributed Configuration concept

The CGS software for checkout and test operations is divided into services for Test Session definition, Test Execution/Control and Test Evaluation.

Central Database repositories provide storage and management of all prepared tables, test results and information supporting the integration activities. In particular a derivative of the Mission Database provides a Test Configuration Database.

The tool design is based on the concept of logical nodes which might be mapped to physical processors in a distributed environment.

There are 3 types of nodes involved in checkout and test operations.

Workstation Nodes

For a Workstation node, the User Interface functions for controlling the execution of Tests are provided. This includes driving of windows and interactive dialogs. This software is available via the CGS Top Level User Interface in particular the Test Set Up, Test Execution and Test Evaluation tasks. (ie CGS Products TSCV, TES and TEV).

In particular, during Test Execution, messages and dynamic display updates are distributed to and from the Workstation Nodes to and from the execution resources.

Test Nodes

For a Test node, the Standard Services required during test configuration/execution such as monitoring, command handling etc, are implemented by the Test Execution Software (TES) product.

There are two types of test nodes:

- Local Test Nodes that provide only local monitoring
- One Global Test Node providing the overall monitoring of the system (MTP).

Database Server Node

This node provides data services such as DBMS, logging/archiving management, printing service etc. The structure of this node is different from the others as it does not have a service requesting part, but only a service provider part. Note that Test Configuration DB and Test Results DB services are provided by Database Server nodes.

Note that a node is not necessarily a separate computer system !

From the allocation of nodes to processors, the following configurations can be derived:

– the **Standalone Configuration**

This configuration comprises one of each node type, i.e. a Workstation, a Test node and a Server node. These nodes reside on two computers: one executing the test node and the other executing the Workstation node and the DB Server node

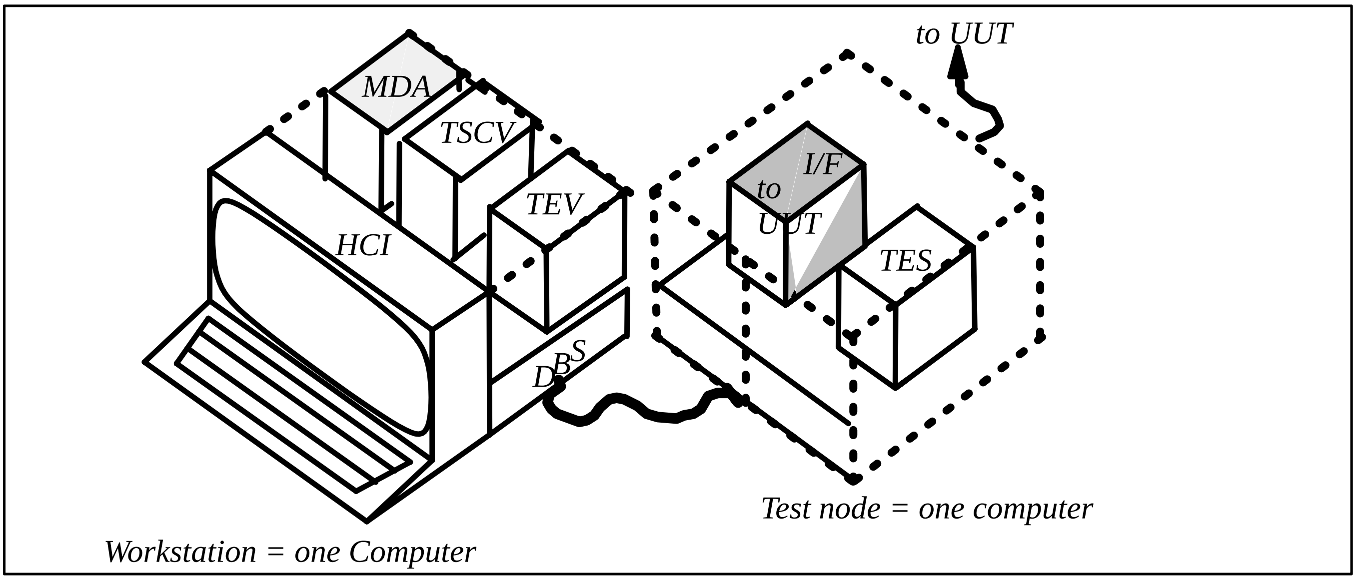


Figure 3–3 : *the Standalone Configuration*

– the **Distributed Configuration**

This configuration includes one DB Server node, several test nodes and several workstations. Each node will then probably reside on one computer, but combining several nodes to one computer is possible.

Note: The combination of two nodes of the same type (e.g. two test nodes) on the same computer is not supported.

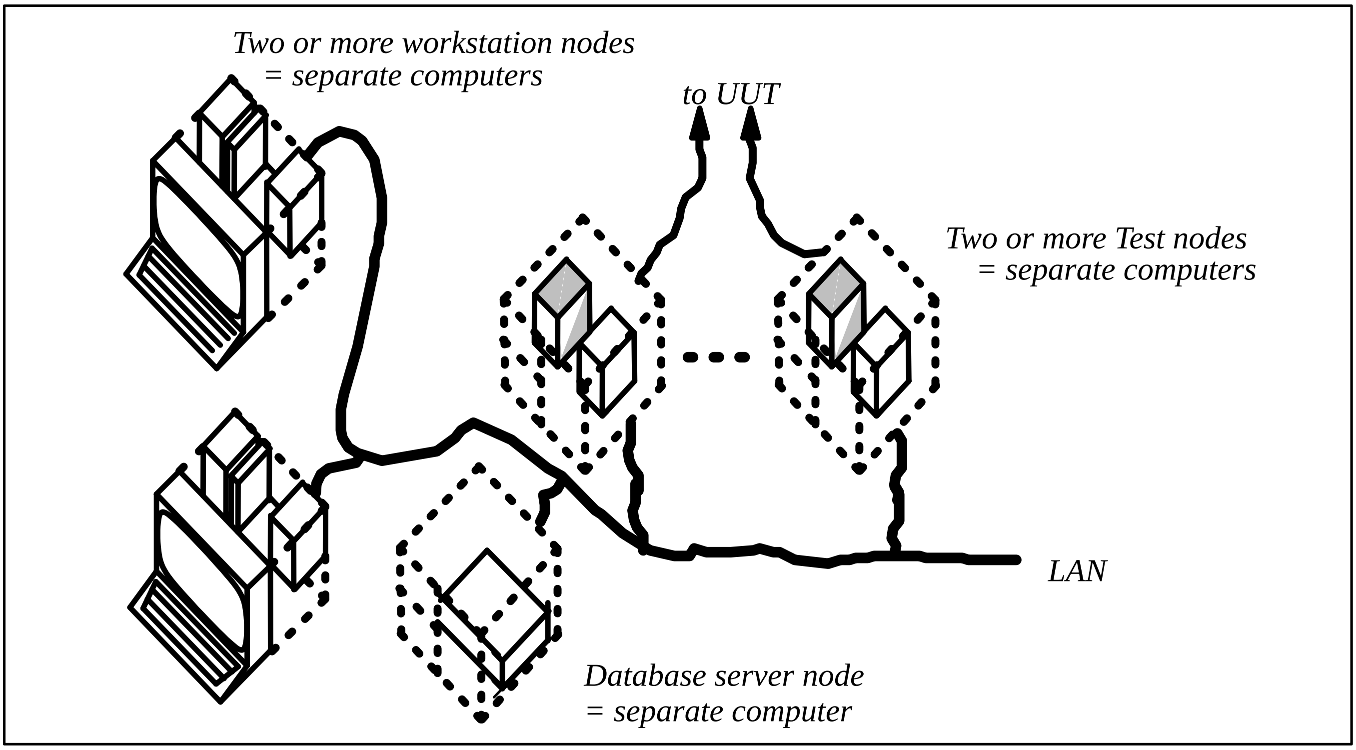


Figure 3–4 : *CGS test software distribution to nodes*

- *A test system comprises a variable number of nodes according to the functional needs of the test – a configuration can comprise 1 to 32 test nodes and 1 to 32 Workstation nodes (On-line).*
- *A test system comprises a variable number of computers according to the performance needs of the test.*

3.2.5 Key feature: the test system architecture is an open system

Among various test objects there will be significant differences in the provided test interfaces. CGS does not aim to include all these interfaces and to implement all the processing required for them.
Instead CGS provides a flexible application software interface that can be used by each User (i.e. SCOE or EGSE manufacturer) to implement the respective interfaces and processing software which runs on the same Node as the CGS software and uses CGS Testing services.

Special Application Software (SAS) constitutes the interface between the checkout and test system and the unit under test (UUT)

- to read data
- to issue stimuli

The checkout and test system provides interfaces for the Special Application Software (SAS) to allow a user to use the monitoring, automatic procedure capabilities and user interface provisions also for data retrieved from or sent to non–standard interfaces.

SAS are separate operating system processes running under CGS control and communicating with CGS via standardised, internal mechanism. The SAS interface is on Ada procedural level.

- *Note that it is a customer task to create the appropriate SAS !*

The special application software allows additional features and allows interfacing of COTS (e.g. EXCEL/ ACCESS).

SAS can read data from the checkout and test system and can provide data to the system for the purpose of special computations:

- to display data in a special way
- to do mathematical transformations
- to store data in a special way

The UCL system library, which is part of the CGS delivery provides a set of predefined modules to ease SAS implementation.

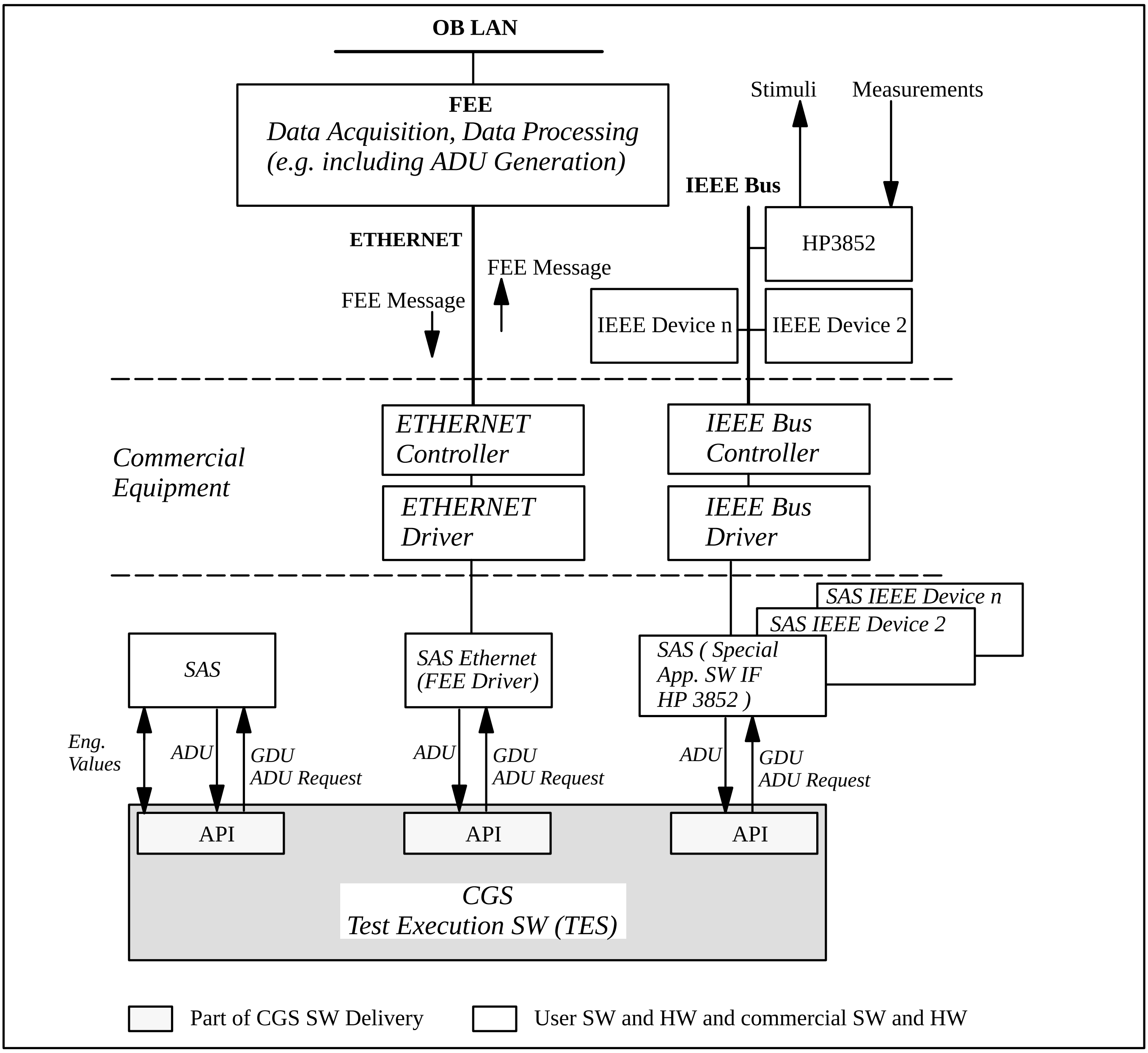


Figure 3–5 : Possible Hardware/Software configuration for a SCOE (test node), indicating the possible extensions of CGS test software to achieve the SCOE tasks

3.2.6 Key feature: various modes of test operation are supported

The test execution activities can be performed in three different ways:

- the **Normal** test execution mode
  in this mode the nominal tests with the UUT are performed.
  - Data is acquired from and sent to UUT
  - All input/output data is archived for replay and evaluation purposes

Simulation mode is normally used for database verification:

- the **Simulation** mode
  in this mode all incoming data is generated within the test nodes itself. The simulation of this

data is driven by predefined data tables. The same functionality as in normal mode except data interface to unit under test is available:

- Input data are simulated (simulated data from unit under test)
- No output to UUT
- For verification of the configuration database and selftest
- Variable timing, i.e. same timing as in normal mode is possible
- Check automated procedures, synoptics and measurement/stimuli definitions

Note that the simulation mode described in this section may not be mixed up with the situation where a simulation model replaces the UUT.

• the **Replay** mode
in this mode the user will see the operations of a previously executed test. The replay submode shall be used to look to events anticipated by the operator which were not encounterd during tests or to investigate situations before test deviations occurred. The data presentation will be in the same way as for normal mode, but all data that are generated to interfaces external to the checkout and test system are suppressed. The replay mode may be set up with different parameters w.r.t. to timing behaviour.

- Replay of previously recorded data i.e. real data from unit under test
- Same functionality as in normal mode except data interface to unit under test
- Variable timing i.e. same timing as in normal mode
- Check automated procedures, synoptics and measurement/stimuli definitions

Replay mode can also be used for database verification.

All modes support the limit checking function, the execution of UCL commands and the visualisation of data.

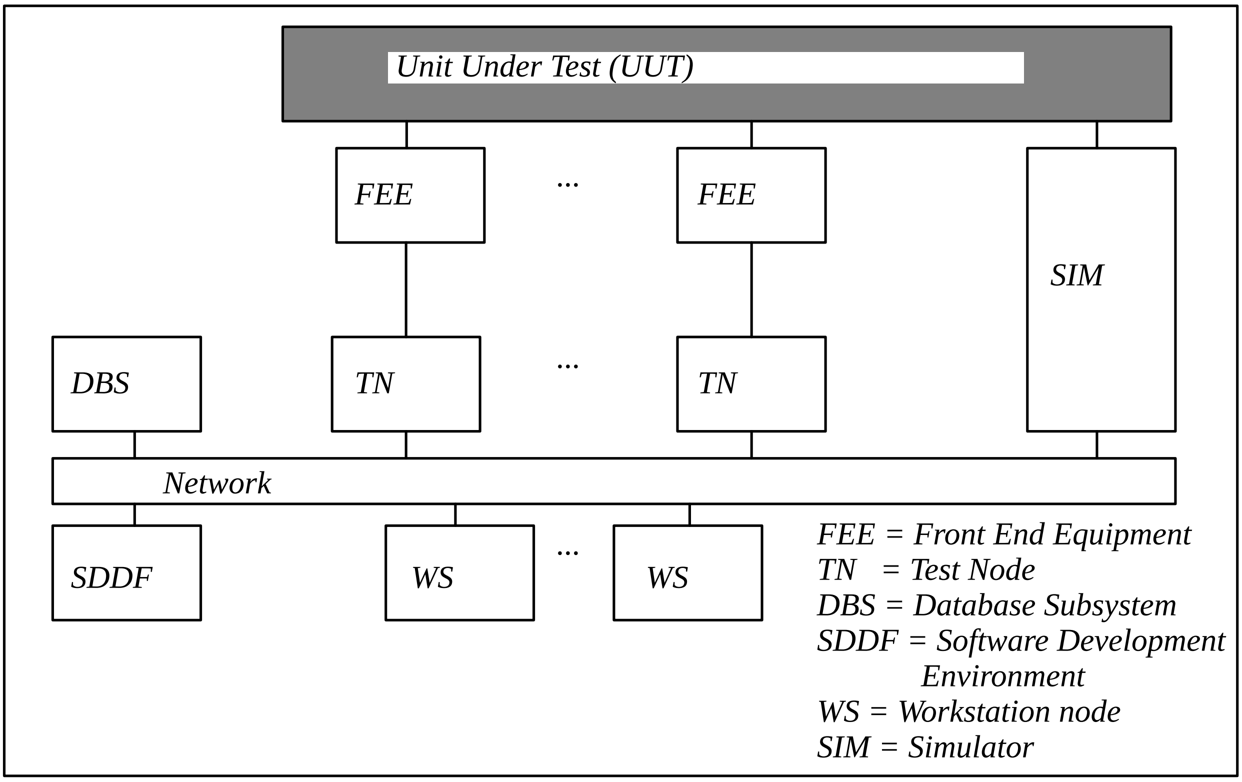


Figure 3–6 : *General structure of a checkout system*

3.2.7 Key feature: Test System Control by dedicated languages

To support SIVQ/AIV operations, CGS Test software uses components of the COLUMBUS Language System (CLS), in particular the UCL (User Control Language) and HLCL (High Level Command Language).

Note that HLCL and UCL have the same syntax !

3.2.7.1 Use of the User Control Language (UCL)

UCL is defined as a set of Test object or Test system oriented commands together with a set of control commands allowing specification of automatic procedures with conditional execution. (UCL is used for both Onboard and Ground purposes).

The UCL Language

- Provides type conversions
- Supports engineering units
- Supports CGS pathname concept
- Conditional statements
  – IF, CASE, LOOP, WHILE, etc
- Libraries
  – user libraries
  – system libraries: GROUND, ONBOARD

UCL allows for definition of Ground Test Automated Procedures (GTAPs) that may be activated by an interactive command and can call other GTAPs. Each GTAP is compiled offline during test preparation by the UCL Compiler. This generates an intermediate language format called i–Code. This is interpreted by the UCL Interpreter called by the Test Execution Software.

UCL supports the calling of general UCL Library routines. Such libraries may be extended by SCOE specific routines to implement SCOE dependent statements in UCL. A call to such a library routine is nevertheless defined in a definition module and imported to the UCL compiler.

The UCL GROUND System Library provides a broad range of function for following topics:

- Monitoring control
  – change online limits
  – enable/disable
  – get monitor status
- Time management
  – get local time and simulated mission time (SMT)
  – SMT management
  – wait
- Automated procedure control
  – start another automated procedure (asynchronous, synchronous)
  – suspend/resume automated procedure
  – get status of automated procedure
  – exchange messages between automated procedures
- SAS control
  – start / stop a SAS
  – change SAS mode (init, reset, etc.)
  – get status of SAS
  – exchange messages with SAS (synchronous, asynchronous)
- Stimulus generation
  – send stimulus
  – send a list of stimuli
  – immediately
  – ”time–tagged”
  – enable/disable certain stimuli
- Event handling
  – generate a log event
  – user events
- Archive & log control
  – enable / disable archiving
  – close the archive
  – enable / disable engineering value logging
- Synoptic display control
  – show a dedicated synoptic on a dedicated screen
  – remove a synoptic from the screen (if owner !)
- User result table control
  – open/close
  –write/read

- User input & output
  – write a message to the user
  – read a message from the user

3.2.7.2 Use of the High Level Command Language (HLCL)

A CGS User involved in Testing operations will be able to issue interactive commands, from the Workstation, to distributed test software via appropriate User interaction methods such as windows, menus, dialog boxes etc. These commands encompass UCL statements and other Workstation or Test specific commands which together form the High Level Command Language (HLCL).

Interpreted keyboard commands issued from the High Level Command Language (HLCL) provide

- single keyboard command
- sequence of individual keyboard commands

HLCL sequences can also be defined by the User (offline) and analysed by the receiving software using the HLCL interpreter. No intermediate compilation is required. The HLCL Interpreter will access the Test Configuration Database (ie instantiation of the Mission Database) to obtain sequences of HLCL commands.

In particular, HLCL supports the invocation of GTAPs in any test node, thus establishing a further level of automation in the system as well as interactive access to each test node.

3.2.8 Key Feature: Test Evaluation Tools for on–line or off–line data evaluation

Test Evaluation software enables the following features:

- Evaluation of data online during test operation,
- Evaluation of data generated in previous tests and for comparison of different test sessions,
- Offline resource (time, processing power, disk capacity etc) intensive evaluation functions,
- Presentation of logging data.

Test Data Evaluation (TEV) is normally performed after having executed an on–line test with the Unit Under Test (UUT) or with simulated parts of it.
The data archived often needs further evaluation in an off–line session, especially to verify wether data generated by the UUT is in the required margins or to analyse non–nominal situations during a test.

On the other hand test evaluation tools can be started to examine the data stored in the test result database during on-line test. This is possible, because data is really written to test result database ”immediately”. The only prerequisite is that the workstation on which the test evaluation shall occur must have visibility (UNIX, NFS, ORACLE) to the database. Even the same workstation that is used for test execution control can be used for test evaluation

Figure 3–7 gives an overview how data are stored and evaluated.

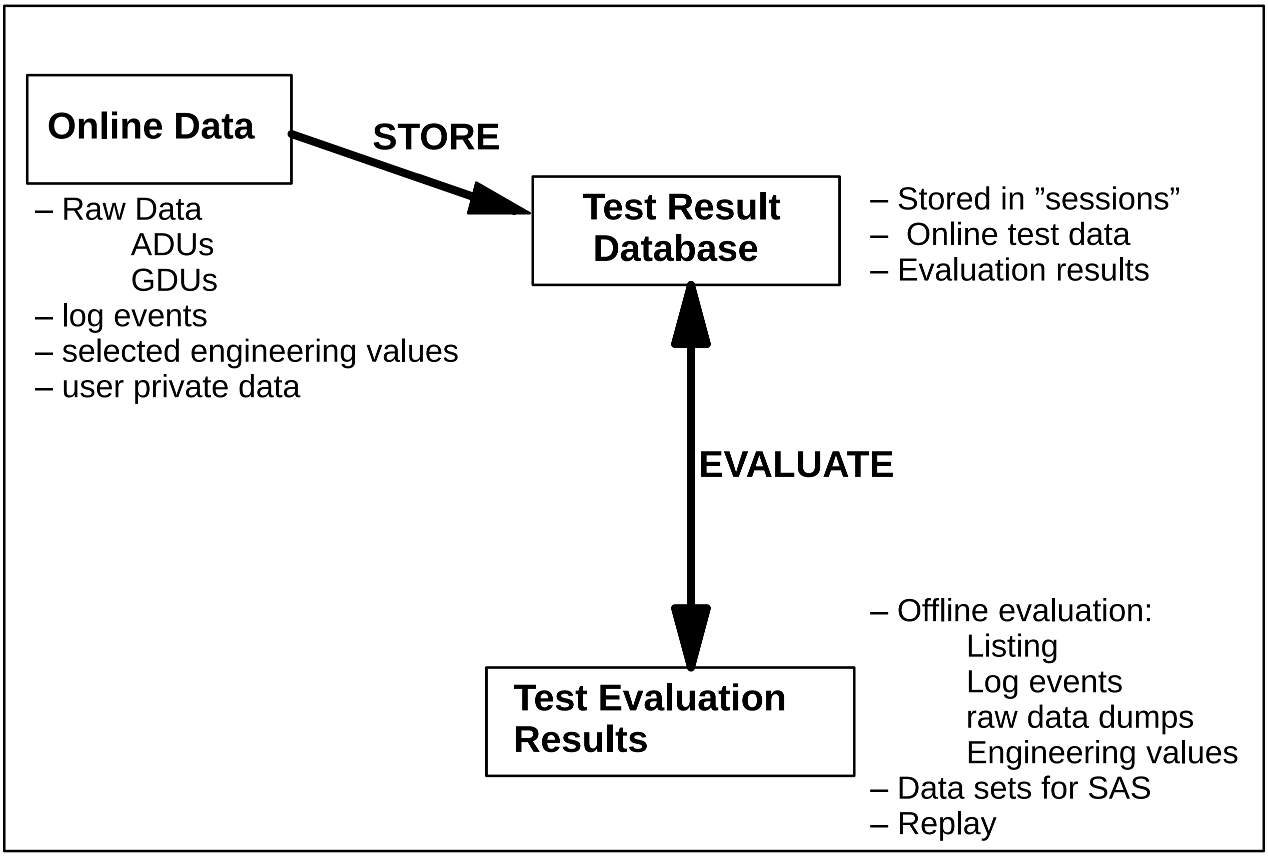


Figure 3–7 : *Data is stored in the TRDB for later evaluation , on–line control or Replay purpose*

During a test following data is archived:

- raw data packets read from SAS
- data packets (stimuli) sent to SAS
- events needed for later replay (e.g. starting SMT)

Archive data are used for test data evaluation and in replay mode

Following data may be logged:

- error situations
- important events (e.g. sending a stimulus, system status changes)
- ”user events”
- engineering values can be logged on user request

All data is stored in the test result database.

Note that all data logically belongs to a so–called ” test session ”.

3.2.8.1 Test Evaluation

In the beginning of a TEV session the user must define the data area in the Test Result Database (TRDB) to be evaluated. This might be a time frame, or simply the name of a test session, specific selection criteria and combinations of them. Additionally the user has to select a CCU version. This may be the same as used during on–line test, but it is also allowed to select an other CCU version.

The user accesses the test result database

- to obtain recorded test data (raw data, data packets, enditem engineering values, log events)

The user accesses the configuration database

- to obtain the enditem’s configuration data

In addition to that the user is also allowed to specify an ”Evaluation Session Name”. Under this name he/she can store all results of his/her work in the TRDB for later review.

TEV tools are available which allow inspection of the data and to select the data according to predefined or specified selection criteria.

TEV provides the following features in this area:

Logging Event Tool
This is the tool for detailed analysis of logged events, such as monitoring exceptions or keyboard commands. It produces listings according to selection criteria provided by the user. The output can be viewed on the screen or saved into result files.

Raw Data Dump Tool
This tool displays raw data fetched directly from the archive files. The user can specify the output format (ASCII/Hex/Dec) for a packet by packet display. The output can be viewed on the screen or saved into result files.
This tool allows the user to perform low–level debugging, e.g. to verify the communication protocol between ground and on–board system.

Data Set Tool
With this tool the user can sample signals from archived data, i.e.
– analog values
– digital values
– string values
The data will be presented with
– the raw value
– the engineering (calibrated) value
– logging time tag
– sampling information (how it was specified by the user)
For such a data collection, TEV offers additional tools for analysis:
– Listing Tool: comfortable layout generator
– Statistic Tool: derives standard statistical results upon data set parameters
– Graph Tool: drawings for data set parameters (line,bar,pie)
– Synoptic Tool: data animation, filling of predefined synoptic pictures with data set parameters

3.3 User Tasks and CGS Configurations

Depending on the user task(s), CGS is available in different configurations. Figure 3–9 provides an overview of the potential user tasks, the corresponding required CGS configurations and the relevant User Manual sections.

User Task	CGS Configuration	CGS Components	Relevant User Manual Section
Onboard SW development	SDDF	CGSI, SDE, SWES, MDA	5.1, 5.2, 6.2
Full mission preparation	SDDF	CGSI, MDA, CLS, GWDU, FWDU, CPL	6.1, 6.2, 7.3
Minimum mission preparation	MBF	CGSI, MDA	6.1
System/subsystem checkout	EGSE	CGSI, MDA, CLS, GWDU, TSCV, HCI, TES, DBS, TEV, NWSW, TSS	4.1 – 4.3, 7.1 – 7.3, 8.1, 8.2
System/subsystem simulation	SITE	CGSI, MDA, CLS, GWDU, TSCV, HCI, TES, DBS, TEV, NWSW, TSS, CSS	4.1 – 4.3, 7.1 – 7.5, 8.1, 8.2
System/subsystem model development	SDDF	CGSI, MDA, CSS(MDE)	7.4, 7.5
Off–line test evaluation	SDDF	CGSI, DBS, TEV, NWSW	9.1 – 9.4
SAS SW development	EGSE/FES	CGSI, SDE, SWES, MDA, CGS_API	7.2 and Appendix F

Figure 3–8 : *User Tasks and CGS Configurations*

4 CGS GENERAL TASKS

Beside the tasks that have to be performed to develop and test a system the CGS user first needs to do some basic activities to setup CGS. These activities are :

- Organise themselves
- Organise other users
- Administration of the work environment
- Error message management
- Free documentation

The following sections describe these activities in detail. The user should first carefully read section 4.1 and if necessary consult the local system administrator regarding the prerequisites prior to using CGS. These prerequisites include a correct installation of CGS as well as the correct setup of the user environment for each user and the correct hardware– and commercial software setup and installation..

4.1 Prerequisites

After a complete installation of the hardware environment, operating system, purchased commercial software and the CGS products, a user may use the CGS products they need to perform the necessary tasks. However, during the installation the administrator has to take some decisions which can affect the operational environment for each user. The following list enumerates these decisions and describes the choices.

- **Operating system installation**
  - The structure of your hardware network

    Check whether your machine is a diskless or dataless client. This is important for data security and performance. A diskless client has all its data on the server. System administration for that server will do a secure backup schedule which could retrieve data lost during a system crash or due to accidental deletion. A dataless client holds your own data on a local disk and will be faster in processing that data.

    You can check whether you have a diskless workstation by the unix–command `df`. Within a shelltool enter `df –k`. The answer will be a list of file systems and their respective mount points within your file hierarchy. The first entry within the table denotes the physical device of the file system. If it is a local disc partition, it will show something like:

/dev/dsk/c0t3d0s0

where dev is the pointer to the devices area in the file system and dsk is a pointer to the disc partitions.

If you find a similar entry in your df output, you have a dataless client, which has a local hard disc. If all your entries start with something like:

hostname:/directory1/directory2...

then all of your file systems are mounted from another machine with the name `hostname` and you do not have a local disc.

Verify where your CGS printer stands. In case you print something with the CGS system you have to find the printouts.

Also verify where a CD–ROM drive, a tape cartridge drive, and a diskette drive is if you ever need to save software to (not in case of CD–ROM, of course) or load software from a respective medium. If you do not find these devices, ask your system administrator.

- The subdirectory which contains CGS and which contains your home directory

Find out where the CGS software is located on your machine. This can be important if you would like to customize your environment and your OpenWindows menu system. You can get the location by inspecting the environment variable `GSAF_HOME`. Therefore within a shell window enter the following:

printenv GSAF_HOME<Return>

The command returns the content of the variable, which should be something like /home–directory. You find out your own home directory by typing into a shell window:

printenv HOME<Return>

This command returns your present working directory, which is immediately after login your home directory. In this directory and all directories below, you have the right to create files, to change actual files, and to delete files. Another way to find out more about your account is to look at your entry in the passwd file, where all information about your account is located, where the system looks what to do if you login. To see your entry in the passwd file, enter the following:

ypcat passwd | grep *YourLoginName*<Return>

The system will respond with one line (or you could say, one record) out of the passwd file, which contains your data. The line comprises entries separated by colons. The entries have the following meaning:

`YourLoginName:YourEncryptedPassword:UserID:GroupID:User-Description:HomeDirectory:LoginShell`

Your login name is the name you enter when you login to the system. Do not worry that anybody can read your password, it is encrypted within the passwd file. The next two entries are translations of your name and group to unique numbers. The user description is a longer description of your name, department and whatever the system administrator enters about you. The next entry is your home directory, that is the directory, where everything belongs to you and where you start your work after you logged in. Finally you see the program which is executed first after you logged in. In CGS, it is recommended to use /bin/csh, that is the C–Shell command shell, which interprets your input as commands and executes them. In the course of starting /bin/csh, OpenWindows will be started, and in each command window, where you are able to enter commands, /bin/csh will be the shell awaiting your commands.

- **OpenWindows installation**

After your first login you see whether your OpenWindows installation is working correctly. The CGS installation of your login procedure will start OpenWindows automatically. If you login the first time, you will get a desktop with one console window, this is a special command tool where all system messages appear, one file manager and its waste basket, where you may perform interactive file access, and one help tool, which gives you an introduction to OpenWindows. Moreover you will have a background menu, if you press the right mouse button when you have the mouse pointer over the background of your desktop. You may manipulate all of these settings and adapt them to your own needs. How this is performed, we will describe in section 4.2. At the top left corner of your screen you see the console window, directly below there is the file manager, at the right side you see the help viewer. The waste basket of the file manager is iconised and hidden behind one of the other windows.

If you click with the right button on the surface of your desktop, that is all the area where no window is located, you get the OpenWindows root menu:

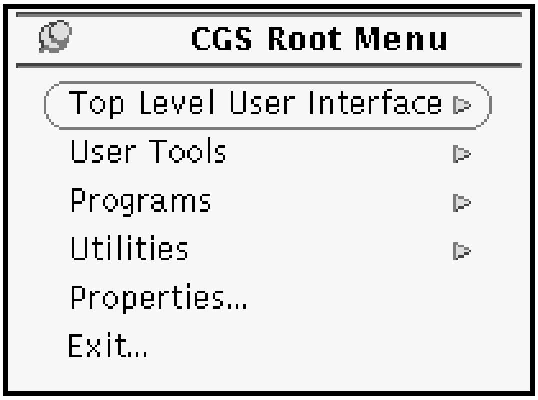


Figure 4–1 : *CGS Root Menu*

If you select one button with a small triangle at the right side, a submenu appears:

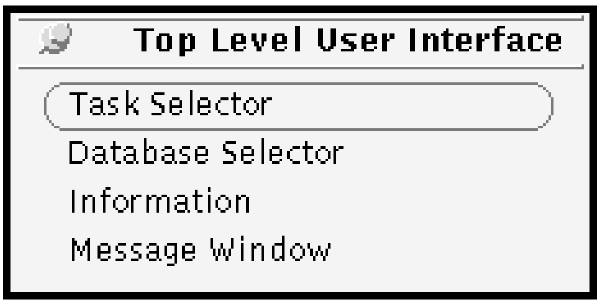


Figure 4–2 : *Top Level User Interface*

The meaning of the menus will be described in section 4.2.

- **CGS installation**

The above mentioned OpenWindows desktop and the menus you get if you click with your right mouse button on top of your background desktop concerns also the correct CGS instal-

lation. If your system administrator correctly installed the operating systems and all related systems, and if he correctly installed you as a CGS user, after your first login you should end up as described above. You can check whether the CGS is installed correctly simply by using it. This is described in starting in section 4.2. However you may verify the correct installation of CGS on your system by inspecting the CGS home directory. Within a command or shell tool enter the following:

Localising Your CGS home directory

- machine–prompt%**printenv GSAF_HOME<Return>**

```
GSAF_HOME=/gsaf_home
```

Listing 4–1 : *Example output for your GSAF_HOME request*

If you now change your directory to that found during the last operation, and make a listing on that directory, you should see something like the following:

Listing the contents of your CGS home directory

- machine–prompt%**cd $GSAF_HOME<Return>**
- machine–prompt%**ls –l<Return>**

```
total 113864
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:25 cgs/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:30 cgsi/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:25 cls/
drwxr-xr-x   7 cgsadmin cgs          512 Dec  2 11:29 cmas/
drwxr-xr-x   2 cgsadmin cgs          512 Jan 23 08:25 config/
drwxr-xr-x   2 cgsadmin cgs          512 Dec 10 15:00 cpl/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 10:58 css/
drwxr-xr-x   2 cgsadmin cgs          512 Jan 23 08:25 data/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:25 dbs/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:25 fwdu/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:25 gwdu/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:25 hci/
drwxr-xr-x   8 cgsadmin cgs          512 Dec  1 04:23 mda/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:26 mdb_ss/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 10:12 mmis/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:26 nwsw/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:26 sas/
drwxr-xr-x   9 dbadmin  develope     512 Jan 23 10:18 sde/
drwxrwxrwx   2 cgsadmin cgs          512 Dec 10 14:16 seq/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:26 smives/
drwxr-xr-x   5 cgsadmin cgs          512 Jan 17 10:32 tdcs/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 10:17 tes/
drwxr-xr-x   5 cgsadmin cgs          512 Jan 23 10:19 tes_sas/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:26 tev/
drwxr-xr-x   3 cgsadmin cgs          512 Dec  3 09:10 tev_sas/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:25 tscv/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:26 tss/
drwxr-xr-x   8 cgsadmin cgs          512 Jan 23 08:26 tts/
drwxr-xr-x   9 cgsadmin cgs          512 Dec  5 16:13 user_home/
drwxr-xr-x   3 cgsadmin cgs          512 Jan 23 16:28 wd/
```

Listing 4–2 : *A GSAF_HOME directory structure of an example CGS installation*

If your listing looks similar to the above, you can be quite sure that CGS has been installed correctly on your system. To verify whether all CGS programs work correctly, look in the following sections on how to use the CGS programs.

4.2 Starting CGS and Task Selection

In the previous section we described which requisites have to be met to use the CGS system and how to verify whether these requisites are fulfilled. You may use your knowledge from that section to start the CGS system. If you login to your workstation for the first time, your system administrator will have established a specific CGS account for you. After login, OpenWindows is started automatically, and you see after a while (a few seconds) your desktop. The desktop is a paradigm which is analogous to the physical desktop in your office. For an example of an OpenWindows desktop directly after a first login, see NO TAG on page NO TAG.

This chapter describes how to start the CGS system after your login to the OpenWindows desktop. Thereafter we will see how you can select the tasks you have to perform with the task selector. At first we will organize the desktop of OpenWindows for better work.

4.2.1 Preparing your Desktop for Work

After you logged in for the first time, you see the help application. Play around a little bit with it. You will learn a lot about OpenWindows, how to use it and how to organise your work.

However, you should create another window where you can enter some UNIX commands. You may choose either a shell tool window or a command tool window from your OpenWindows background menu, as shown in Figure 4–3 below.

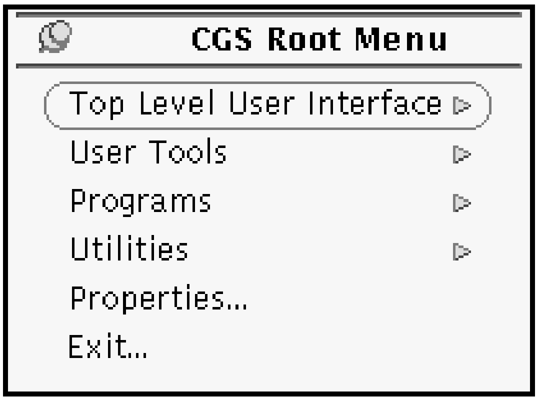


Figure 4–3 : *CGS Root Menu*

The OpenWindows background desktop menu can be configured by the user. It resembles the information given to it from the file `.openwindows-menu` in your home directory. See an example of a configuration file in the following Listing 4–3:

```
”CGS Root Menu” TITLE

”Top Level User Interface” MENU
    ”Task Selector”              exec Start.Task_Selector
    ”Database Selector”          exec Start.Database_Selector
    ”Information”                exec Start.Information_Utility
    ”Message Window”             exec Start.Message_Window
”Top Level User Interface” END PIN

”User Tools” MENU                .user/rootmenu.User_Tools

”Programs” MENU
    ”Command Tool...”            exec cmdtool.CGSI
    ”Text Editor...”             exec textedit
    ”File Manager...”            exec filemgr
    ”Mail Tool...”               exec mailtool
    ”Calendar Manager...”        exec cm
    ”Clock...”                   exec clock
    ”Calculator...”              exec calctool
    ”Print Tool...”              exec printtool
    ”Tape Tool...”               exec tapetool
    ”Binder...”                  exec binder
    ”Snapshot...”                exec snapshot
    ”Icon Editor...”             exec iconedit
    ”Performance Meter...”       exec perfmeter
    ”Shell Tool...”  DEFAULT     exec shelltool.CGSI
    ”Dbx Tool...”                exec dbxtool
”Programs” END PIN

”Utilities” MENU
    ”Refresh”  DEFAULT           REFRESH
    ”Restart”                    RESTART
    ”Reset Input”                POSTSCRIPT /resetinput ClassUI send
    ”Save Workspace”             SAVE_WORKSPACE
    ”Lock Screen”                exec xlock
    ”Console...”                 exec cmdtool –C
”Utilities” END PIN

”Properties...”                  PROPERTIES

”Exit...”                        EXIT
```

Listing 4–3 : *An example .openwin–menu configuration file*

Note that you may configure the menu only in the **User Tools** submenu area. To achieve that, edit the file `rootmenu.User_Tools` in the directory `.user` in your home directory. For the syntax and the meaning of the contents of the listing refer to your OpenWindows documentation.

But before you can work with OpenWindows menus, let us see how to select something from the menus. There exist two basic selection techniques:

- Press–move–release
- Click–select

Both techniques may be used in a nested way. You always select a menu of choices in OpenWindows with the right mouse button.

4.2.1.1 Press–move–release technique

Now if you press the right mouse button on your OpenWindows desktop background and keep it pressed, the CGS Root Menu shown above will appear (or at least a very similar looking menu). The only difference with this technique will be that the push pin button you see top left of the menu will be pulled out. Now, as long as you keep the right mouse button pressed, the menu selection will be active. If you now drag the mouse pointer along the menu, you will see that some choices of the menu will become inverse (that is white writing on dark background). One of the choices will be surrounded by a fine line. This choice is the default choice of this menu. On the top level this makes no sense, but on submenus this will be chosen if you select a choice of a highlevel menu without looking into the submenu.

If you release the right mouse button on a choice while it is highlighted (reversed), you select this choice. If you release the button on any other region when no choice is highlighted, you choose nothing and stop your menu selection. Anyway the menu will disappear after you release the button.

Note that all processing of the OpenWindows window manager is interrupted during a background menu choice. It will look like the machine is frozen. But that is only on the surface as all background processing will continue.

Now look at the small triangles at the right side of some of the choices. If you move the mouse pointer over them targeting right, always with the mouse button pressed, you will realize a new menu box springing off the first one. This is called a submenu, and you may have a large hierarchy of submenus. It is an OpenWindows standard that choices which will start a program have three periods at the end.

Play around a little bit with this menu selection policy. You always select a choice by releasing the right mouse button. If you release the mouse button on a menu which would have lead you to a submenu, you select the default choice out of this submenu.

4.2.1.2 Click–select technique

If you press a mouse button and release it after a well defined short time, this is called a click. You can specify the period of time in an OpenWindows configuration file for each user. Moreover you may have a double click, that is two clicks in a well defined period of time. You also have in some application a triple and quadruple click. For our background menu we need only a single click.

If you click with the right mouse button on the OpenWindows desktop background, you will get the same menu as with the other selection method, but now, you may draw around with your mouse pointer and need not to keep anything pressed down during that action, and the menu box will be kept. You may even do other things. If you come back, the menu box will still be there. The surface of the machine will be frozen. This technique releases you from keeping the right mouse button pressed all the time.

If you now click on a choice which is a submenu, the submenu will come up the same way. This may be continued until there are no more submenu choices. You can select a choice with the left button. If you use the left mouse button on a submenu choice, you will select the default out of this submenu. If you use the right mouse button, you will get the submenu box. You may change your technique from click–select to press–move–release technique but not vice versa.

There is one speciality which we already mentioned above: the push pin buttons. If you see a push pin button on the top left side of the menu box, which is currently pulled out, you may select it or click on it, dependent on your choice technique, this push pin button will be pushed in. After that, the menu will be permanent and you can use it any time you like. After the menu is permanent, the screen will no longer be frozen. With this means you may create an own working desktop, where you have your menus permanent under access.

4.2.1.3 Creating a command tool

Now let us go back to our initial task of creating another tool for command line interface to UNIX. As mentioned above you may use either a

- command tool

or a

- shell tool.

The difference between the two is only on the surface. The command line interaction is the same and depends on your command shell. Examples for a command shell are the C–Shell, the Korn–Shell, or the Bourne–Shell. Within a shell tool all your inputs and the responses of the system will disappear at the top border of the window. Within a command toll you will have a scroll bar at the left or right side, and all system interaction will be kept. This comes at a price, for it costs you resources in the form of memory.

For most people it is a matter of taste which they prefer. Since the systems of today have large amounts of recourses, many people prefer a command tool because they always can go back to in– and output which has been done before and inspect it or even reuse it. So now we will start a command tool with the OpenWindows background menu. In the following, A menu selection will always be shown by the path through the menu hierarchy: selection1=>selection2=>selection3. This means you select from menu 1 selection1 which is a submenu. From this submenu you select selection2 which is again a submenu, from which you select selection3 which is the final choice.

Now, from the OpenWindows background menu, select Programs=>Command Tool... After a while, a new window will pop up. This window looks similar as the CONSOLE window and has the same kind of prompt waiting for input of you. You may position and resize the window as you like. For more information on how you can handle your OpenWindows desktop and windows, inspect the help tool which comes up after log in.

Creating a command tool window

- **Background–menu:** Programs
- **Programs–submenu:** Command Tool...

If you would like to make your current desktop configuration permanent, there is a possibility to save your desktop: From the background menu, select Utilities=>Save Workspace. This will last a short time. After the save has successfully finished, there comes a small confirmation window with the text **Save Workspace Complete** and a small button tagged with **ok**, where you have to click to confirm the action.

Saving your desktop workspace

- **Background–menu:** Utilities
- **Utilities–submenu:** Save Workspace

Your changes are saved in the file `.openwin–init` in your home directory. You must not edit this file manually.

4.2.2 Starting CGS from OpenWindows Desktop

In your OpenWindows background desktop menu there is a choice called **Top Level User Interface**, which has been included for you by CGS. If you select this choice, you will get a submenu which looks as shown in Figure 4–4 below. You may push in the push pin button to make the top level user interface application permanent.

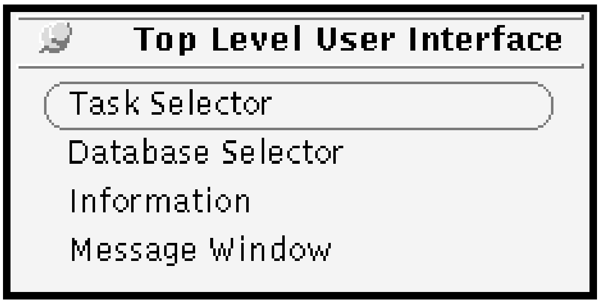


Figure 4–4 : *OpenWindows submenu **Top Level User Interface***

We will now examine the above depicted submenu and the four choices.

4.2.2.1 Task Selector

The choice **Task Selector** will start two applications: The first one is the so called welcome screen. It shows you some information about the CGS and the company who developed it. It is a visual reminder for you that the CGS now has been started. You may click on the **Continue** button at the bottom of the window. An example of the window is shown in Figure 4–5 on page 4–10. Note that you may work with CGS even if you do not press the **Continue** button. In that case you probably would like to minimize the welcome window to an icon. You do that by choosing from the top bar menu from the welcome window the selection **Close** or by pressing the Open key on your keyboard while you are within the welcome window with your mouse pointer. You get the top bar menu of any OpenWindows window by moving the mouse pointer to the top bar of the window and then pressing the right mouse button. Here you can use the same selection policy as for the OpenWindows background menu as depicted above in section 4.2.1.2 on page 4–8.

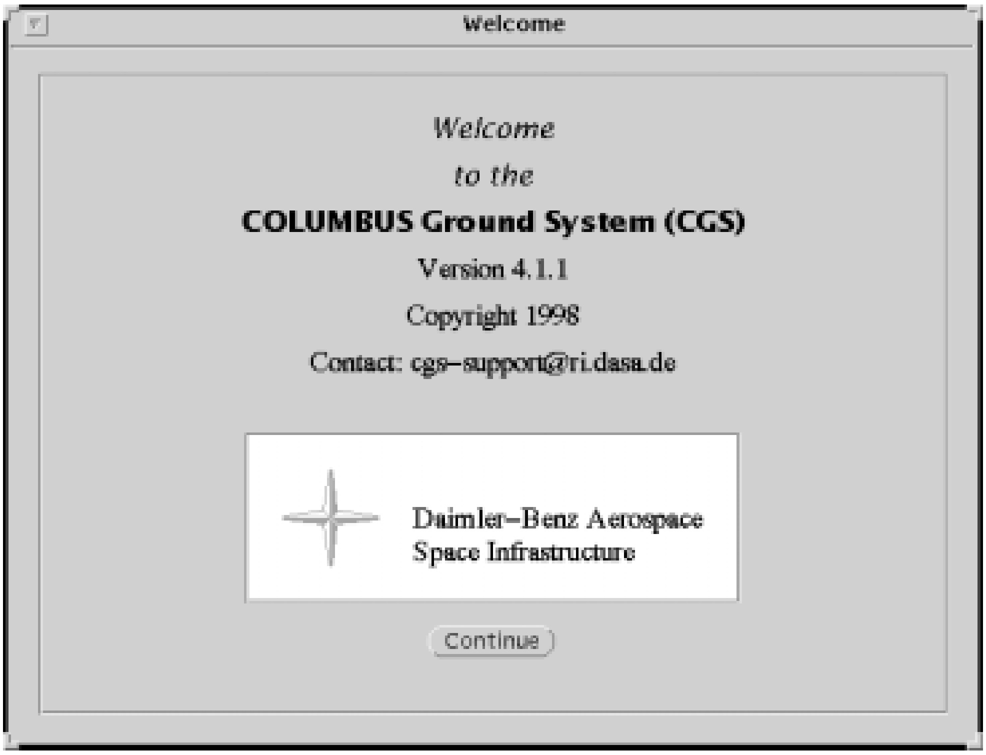


Figure 4–5 : *The CGS welcome window*

The second application which comes up after you select the button **Task Selector** from your submenu **Top Level User Interface** is the **CGS Task Selector** window itself. The task selector window is shown in

Figure 4–6 below. The task selector is the starting point in an own CGS menu system which guides you through the CGS applications. It will be described in section 4.2.3 on page 4–13 and the following pages.

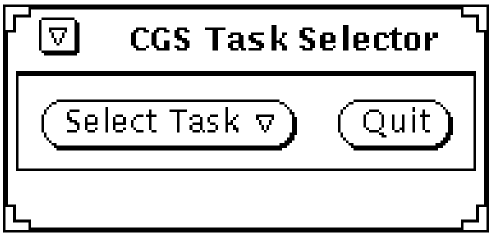


Figure 4–6 : *The CGS Task Selector*

4.2.2.2 Database Selector

With the second choice in the Top Level User Interface menu you start the database selector. In a networked environment you may have access to several databases at a time. To show all your CGS applications which database you are using currently, with this application you can select where your database requests will be headed to. In your hosts database of the Network Information System (NIS) of the SunOS, there will be one host which has the nickname dbhost. Normally all your database requests will be sent to this host. But with the database selector you can choose to point to another database. The database selector window also displays the current database. If the database is local, there will be no networking at all, according to databases, but the database requests will be processed on the host where you are currently logged in.

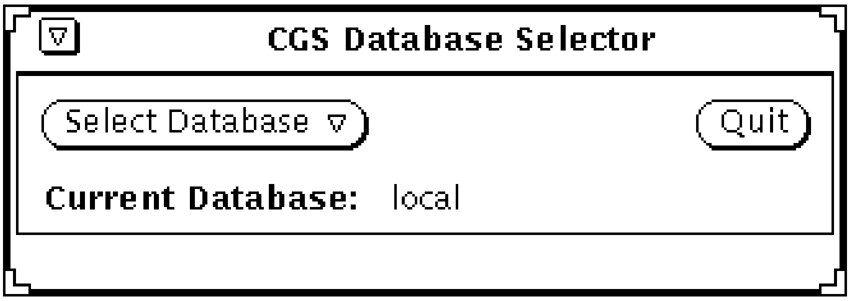


Figure 4–7 : *The CGS Database Selector*

If you selected a new database location, the database selector application does not need to remain active. You may exit the database selector application, because it will change the pointer to the correct database in your environment. This change will be permanent, that is, it will be there even if you reboot your machine.

To configure things, the administrator has to prepare the file $GSAF_HOME/cgsi/data/databases. It shall contain one line per database instance like the following:

```
"  Database 1  "  oracle.world
"  Database 2  "  oracle2.world
```

Each line contains a label and the TWO_TASK environment variable. After setting the value, applications need to be restarted to take the change into account.

Note, that applications living for a longer time may cause problems.

4.2.2.3 Information

With the button **Information** you start an application which displays an information window as depicted in Figure 4–8. The contents of the window serve only to give you the following information:

- **CGS version**: this is the overall version number of your CGS. Please note that it is not necessarily the version of any piece of software below CGS
- **Node name**: this is the workstation where you are logged in at present
- **User name**: this is the Unix user name of your current login at the workstation where you are logged in

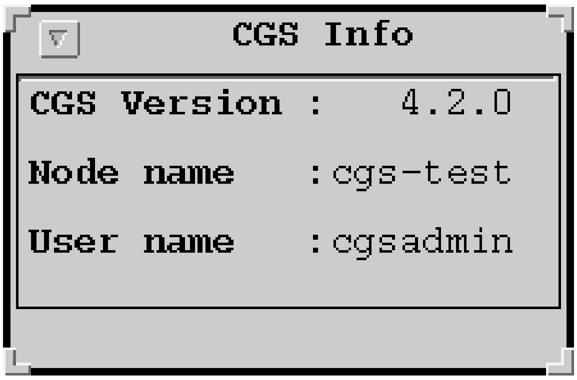


Figure 4–8 : *The CGS Information*

There is no further interaction with this window.

4.2.2.4 Message Window

The button **Message Window** starts the error services application, which gives you a focus on all error messages issued from the CGS applications. You have a lot of possibilities to fine tune the reporting of error messages. A complete description of the message window utilities will be given in section 4.3 about error handling. The application window you activate by clicking on the button **Message Window** is shown in the following figure:

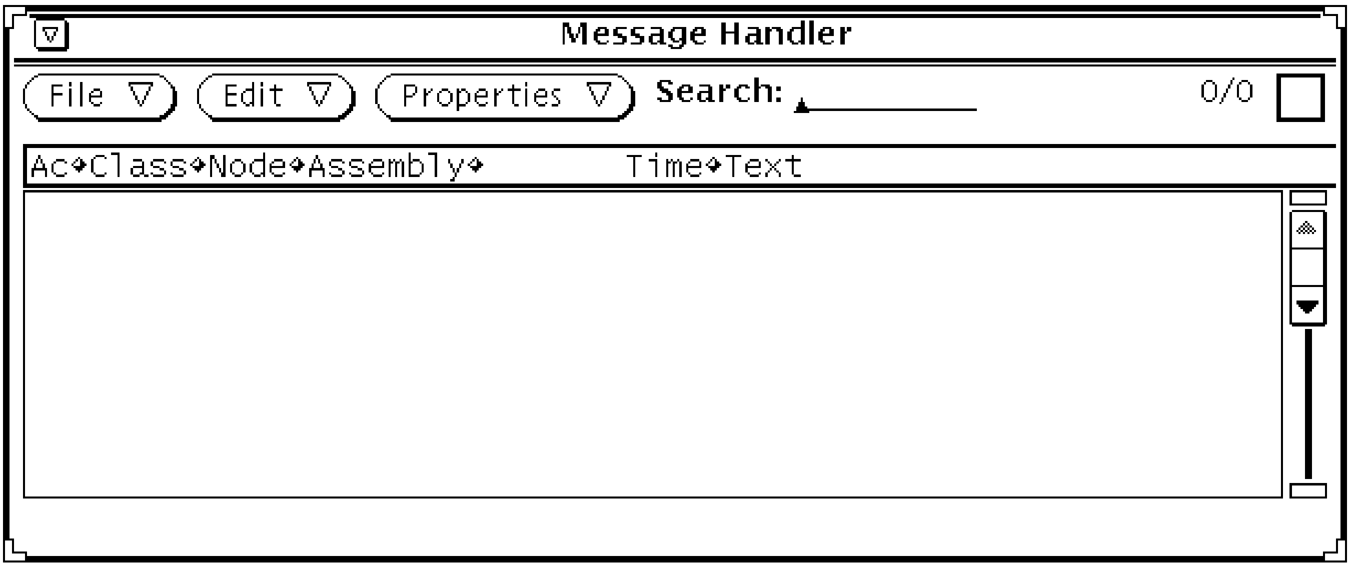


Figure 4–9 : *The Message Handler Window*

You should keep this application active to get an overview on the messages the different CGS applications will send to you. If you select the entry ”Info ...” from the file menu you will see some information of the CGS version, user, machine and the CGS e–mail address. Contact this address in case of problems.

4.2.3 Task Selection

In a standard CGS environment you may select applications or commands from a configurable list of ”tasks”.

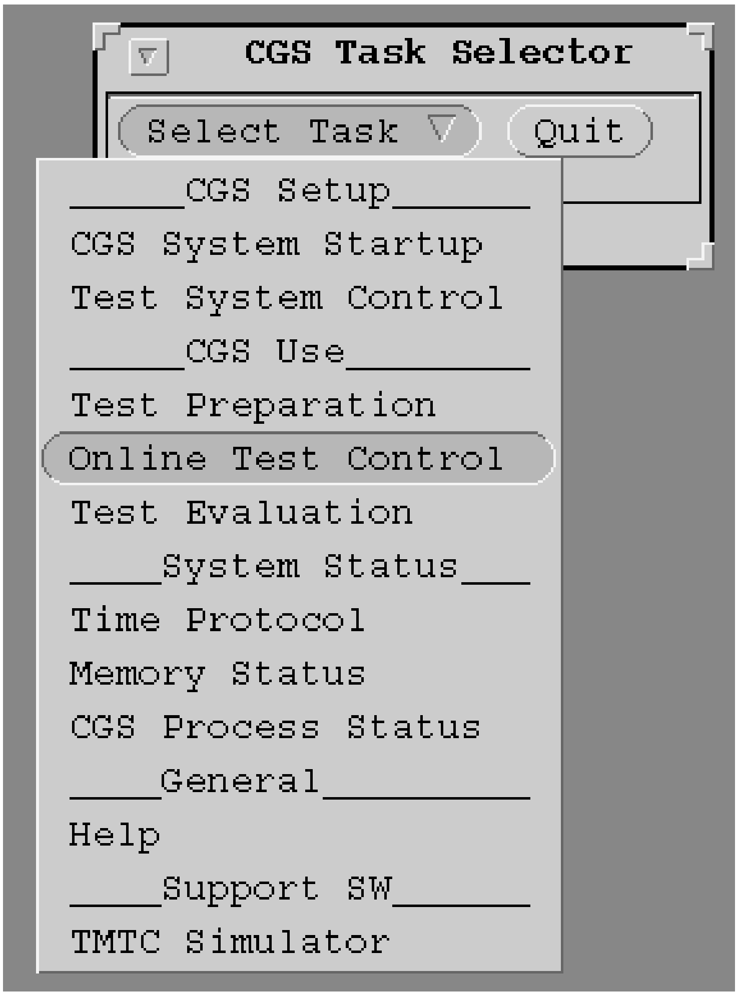


Figure 4–10 : *Figure Figure 4–11*

The Task Selector is configured by a file in your home directory. This is the file `.task-list`. An example of such a configuration file is depicted in Listing 4–4 below. The column on the left edge represent the entries in the menu which you see, after you click with the right mouse button on **Select Task**. The right column contains the corresponding application which is called by the system after you selected the menu choice. The task selector application needs to find the corresponding application, for example **I_MDB**, in your search path.

It is possible to change the contents of the file `.task-list`. Instead of single application names you may include complete pathnames which point to the application in question. If the task selector application does not find an application or there is an error starting it, the error message will be sent to your CONSOLE window.

```
"_____CGS Setup______"     DUMMY    (remember: no tabs)
"CGS System Startup"   $CGS_HOME/bin/common/start_cgs
"Test System Control"  $TSCV_HOME/bin/common/start_tscv
"_____CGS Use________"    DUMMY
"Test Preparation"     $MDA_HOME/bin/common/I_MDB
"Online Test Control"  $HCI_HOME/bin/common/start_hci
"Test Evaluation"      $TEV_HOME/bin/common/start_tev
"____System Status___"    DUMMY
"Time Protocol"       $CGS_HOME/util/common/Time_Protocol
"Memory Status"       $CGSI_HOME/util/common/memory_status
"CGS Process Status"  $CGSI_HOME/util/common/CGS_processes
"____General_________"    DUMMY
"Help"                  Start_Help_Facility
"____Support SW______"    DUMMY
"TMTC Simulator"       $GSAF_HOME/tts/bin/sun5/tts
```

Listing 4–4 : *An example task_list*

4.3 Error Handling

4.3.1 Intro

At any time things may go wrong and this information has to be forwarded to the operator. If interactive commands or controls fail, this information should come right back to you using the appropriate interface. ASCII consoles should print an error line, graphical controls should beep and show some indication wherever appropriate.

There are however situations, where errors cannot be mapped back to a direct action or producer. The system needs to log these information and provide it to the operator by some means. How you can handle this information is the topic of this chapter.

4.3.2 Concept

All CGS assemblies may deliver error messages, warnings and other information using the CGSI error service. These messages are logged on each host by a ”Message Server”–process. This process is part of the CGS startup and shutdown, so it is always available on an up and running system.

These messages may be viewed with the Message Handler. While operating the CGS system this application should be always activated to keep you informed about the ongoing tasks and error events.

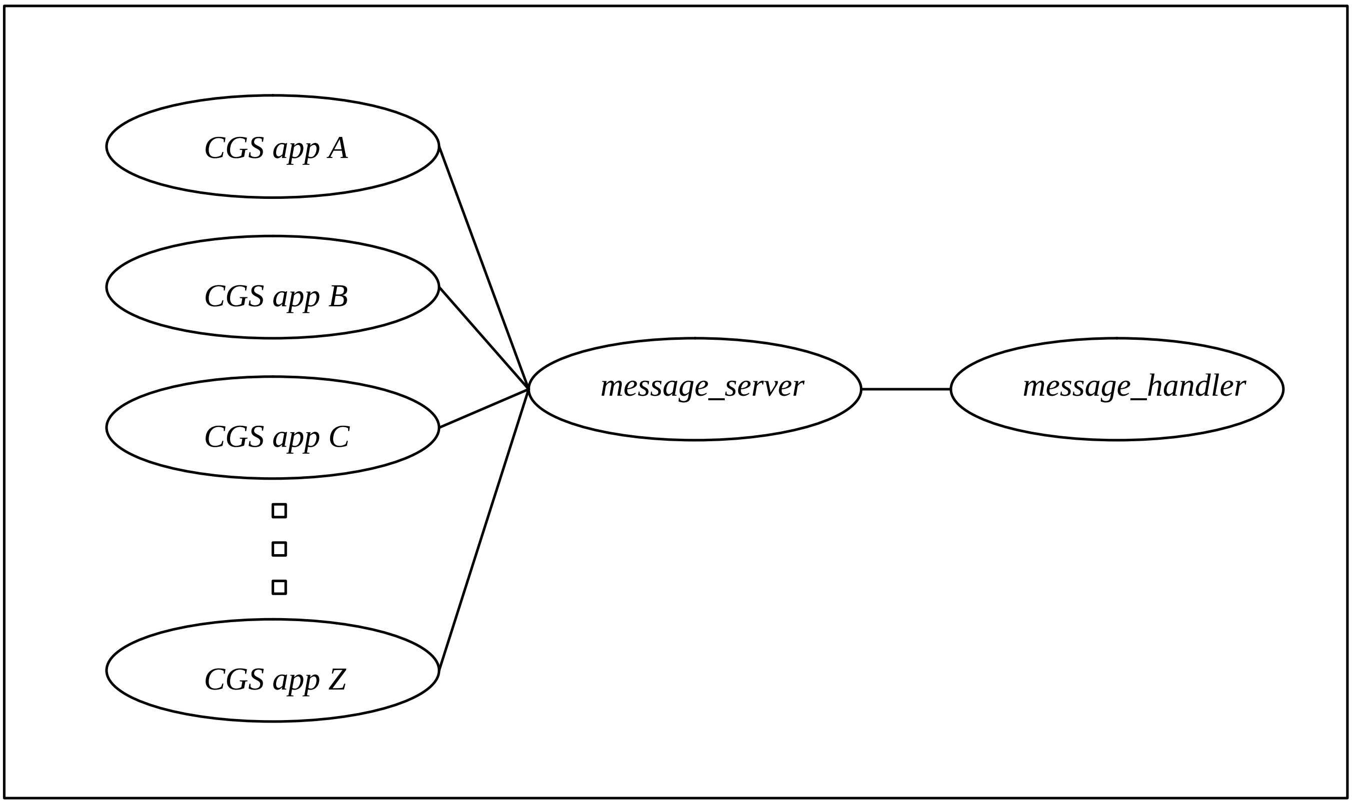


Figure 4–12 : *Error service process communication*

4.3.3 Starting the Message Handler

As already described in section 4.2.2.4 on page 4–12, you start the Message Handler window by selecting the choice **Message Window** from your **Top Level User Interface** submenu out of your background Open-

Windows desktop menu. For an example of your Top Level User Interface submenu see Figure 4–4 on page 4–10.

Starting the Message Handler window

- Move the mouse pointer to a free part of the desktop.
- **Press** the right mouse button. The **CGS Root Menu** appears.
- Select **Top Level User Interface–>Message Window.** The message window appears in the lower part of the desktop.

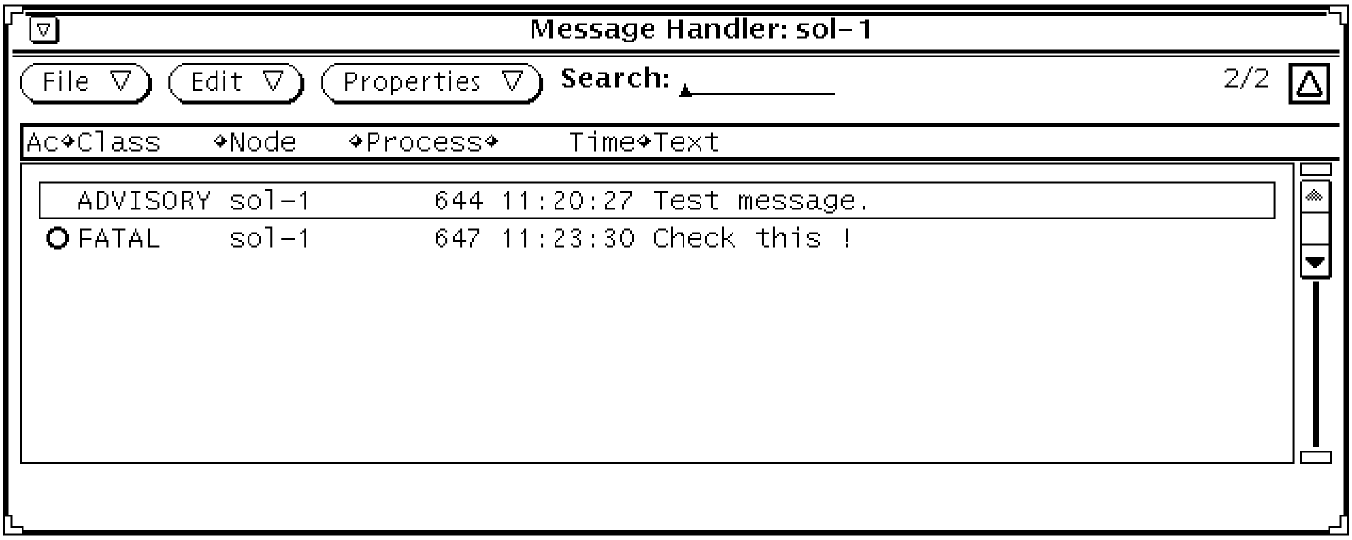


Figure 4–13 : *The Message Handler Window*

At startup, the Message Handler tries to connect to the local host. A successful connection is indicated by the hostname being added to the frame label (here ”sol–1”) and by the warning triangle in the drag–and–drop target at the top right corner.

If the connection fails, a beep can be heard, a warning is displayed in the left footer and no connection is indicated neither in the frame header nor in the drag–and–drop target.

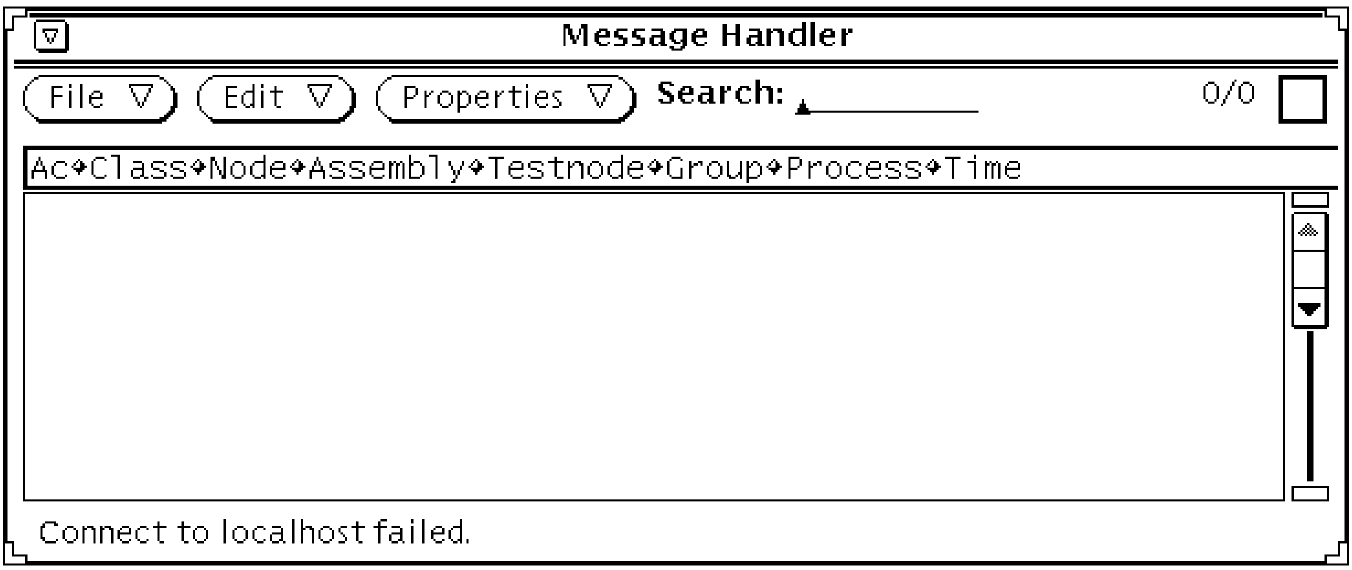


Figure 4–14 : *The Message Handler Window with failed connection*

4.3.4 The Main Window

The Message Handler displays the tool name plus the data info (connection or filename) at the top of its main window. In the left footer some status messages and warnings are displayed.

There is a list of the following menu buttons:

- **File**
- **Edit**
- **Properties**

There is a text field for searching a string in the list of messages.

- **Search**

  Just enter some text and type RETURN. The message containing the text will be selected. Note, that you can configure this search to search through the complete content of each messages or just the text displayed in the list, see the end of chapter 4.3.8.

Note, that the search is case sensitive, still.

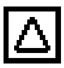
Due to configurations you see a subset of the messages received in the system. Numbers indicating the subset are displayed near the top right corner.

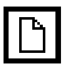
- **displayed / received** messages

  Read it as ”Displayed are X of Y messages”. You may configure your Message Handler to see only special kinds of messages, see chaper 4.3.8 for details.
  Further, the applications sending messages in the system may specify hosts and users that should see a message. So it may be the case, that some messages from a special daemon are only visible to an operator at that local machine.

In the top right corner you see a”Drag and Drop”–target. This is an OpenLook metapher for an area where you can copy data from or to by dragging with the left mouse button.

- **Drag and Drop** target

   an online connection to a server is established

   a file is being viewed offline

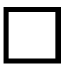
   no connection, no file

Figure 4–15 : *The drag and drop target*

Below the menu buttons you see the list of messages.

- The **message display area**

  This area shows a scrollable list of messages that have been received. The messages are displayed according to the setup defined in the **Properties –> Tool...** window. Each message field displayed is indicated in the header line. See chapter 4.3.8 on how to configure this display.

By double clicking on a list entry the message property window pops up showing all the fields of this message in detail.

4.3.5 The File Menu

The file menu contains the main entries for the tool. The I/O with the other applications or the file system is provided here.

At any time, the user may

- get some info
- connect to some Message Server
- disconnect from a Message Server
- open some logfile, thus closing a possible connection to a server
- save messages to a logfile, leaving a connection or opened file untouched

Figure 4–16 : *The Message Handler file menu*

When ”Connect ...” is choosen, the Message Handler looks for reachable Message Servers. After waiting 5 seconds for replies the list is displayed.

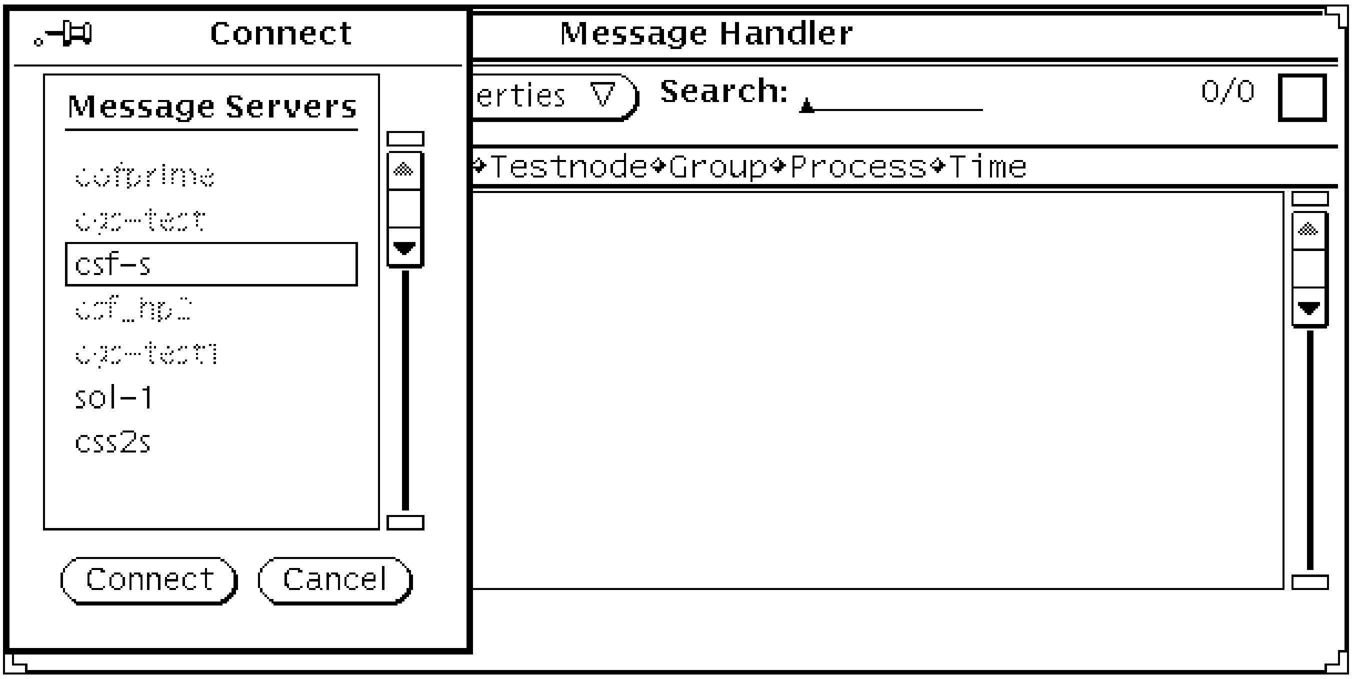


Figure 4–17 : *Connecting to a Message Server*

All active Message Servers in your CGS cluster will be provided. Note, that other servers from your network neighbourhood might answer, too. This depends on your network configuration for broadcasting. In the Figure 4–17 above, these servers are grayed out, since they answered to the broadcasted request, but their logfiles are not available for this Message Handler.

When choosing ”Open ..” or ”Save ...” from the File–menu, a file selector box is displayed, where you can choose the name of a logfile.

If you are viewing old logfiles, you can apply masks and all features provided by the Message Handler, there is no restriction for this offline investigation. You can analyze problems of special systems or hosts, search for text strings and so on.

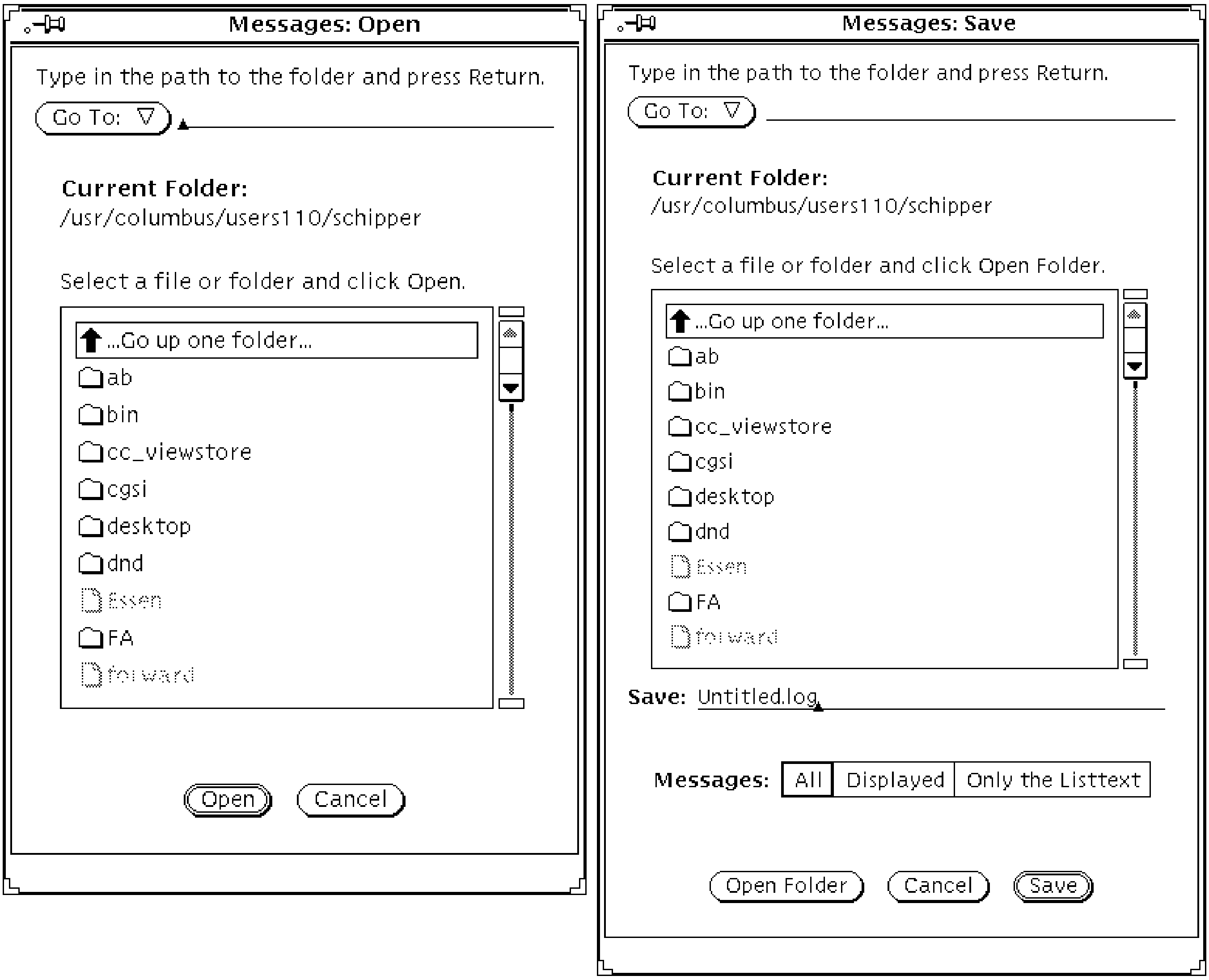


Figure 4–18 : *Opening / saving a message log file*

If you have older logfiles from CGS version prior to 4.1.1, you can convert them using a small utility.

Convert pre–4.1.1 logfile

■ prompt% convert_log < old.log > new.log

If the converter cannot be called just like this, look for it as $GSAF_HOME/cgsi/bin/sun5/convert_log.

Note however, that some information is missing in the old format and replaced by dummy information:

- aimed user
- aimed host
- application required ACKNOWLEDGE
- handling (LOG_ONLY / LOG_AND_DISPLAY)

The old implementation just acted to these settings and did not put this info into the log file.

The producers of messages may specify a target host and a target user. Most applications want to send their messages to any user and any host.

But some applications may specify the local host for performance or a dedicated administrator user because nobody else could solve some local problem.
In this case, you will not see all messages online, no matter what you specify with the Message Handler Properties.
In offline mode, i.e. when evaluating old log files, the Message Handler shows ALL messages, no matter what the address was thus giving complete access to the log
So you may see more messages offline, than you saw online. This is no bug, it is a result of the concepts.

4.3.6 The Edit Menu

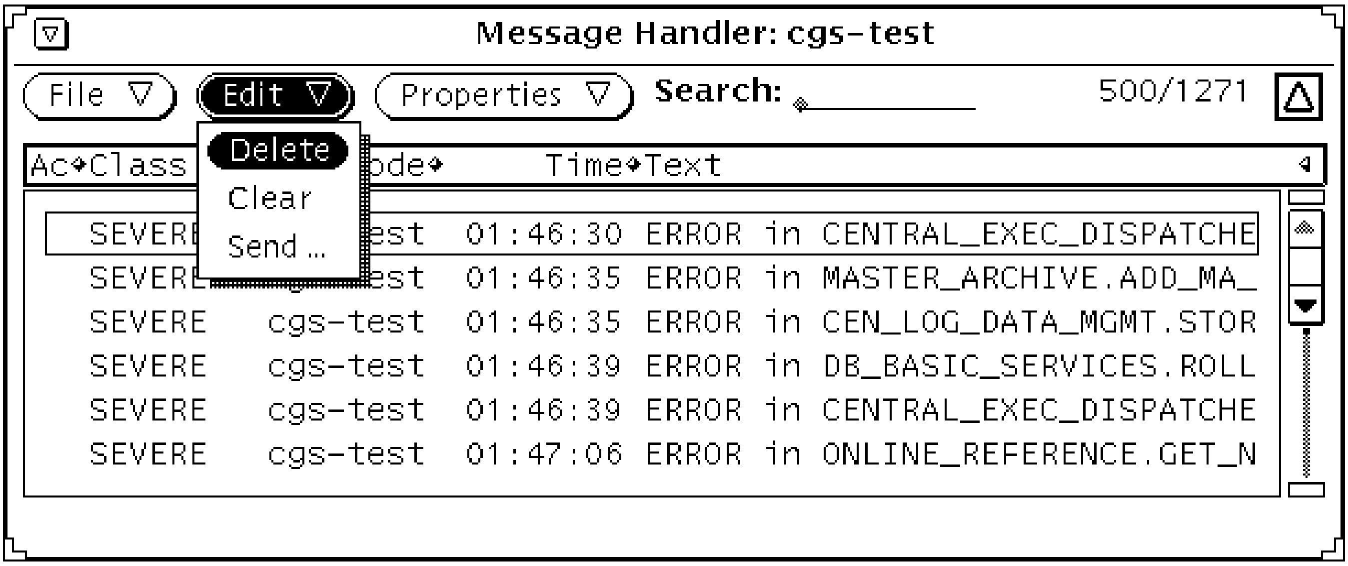


Figure 4–19 : *The Edit Menu*

You may

- **Delete** a single selected message
- **Clear** the display

  Note, that all messages are still logged and displayed again by reconnecting to the server.

or

- **Send** a message.

Sending a message should be done with care. During normal operation, it should not be necessary. In strange or critical situations however, it may be the proper medium to inform all operators. It is the simplest way to test the messaging in general and the configurations in detail.

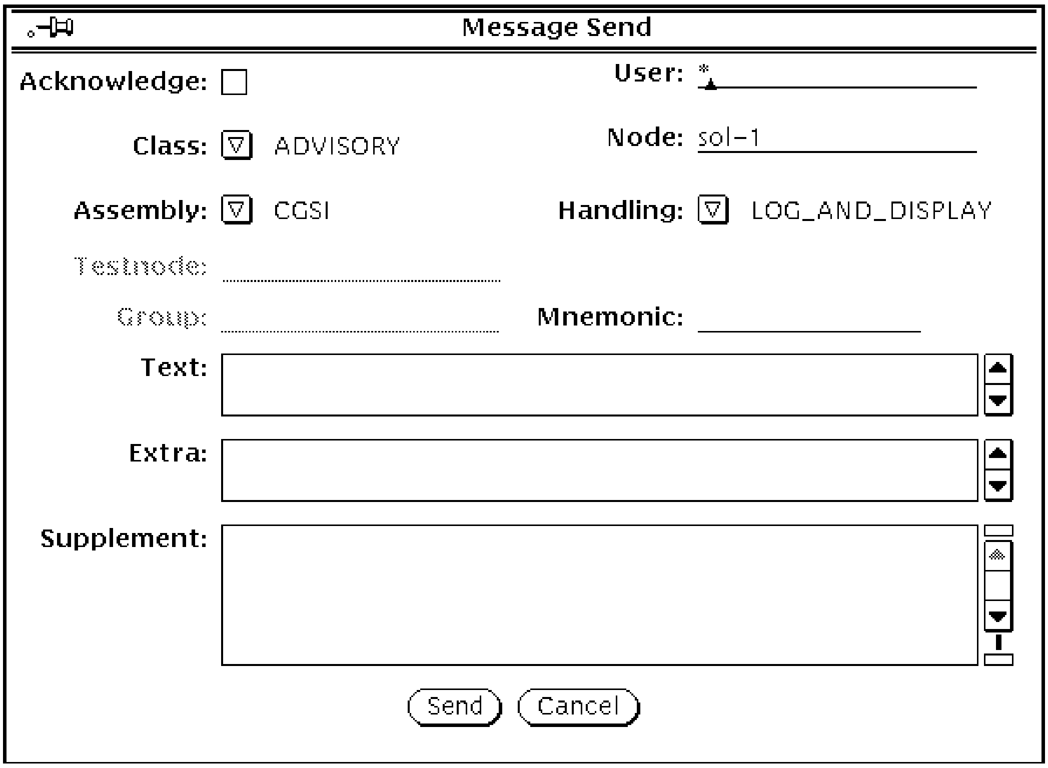


Figure 4–20 : *The Message Send Window*

4.3.7 The Property Menu

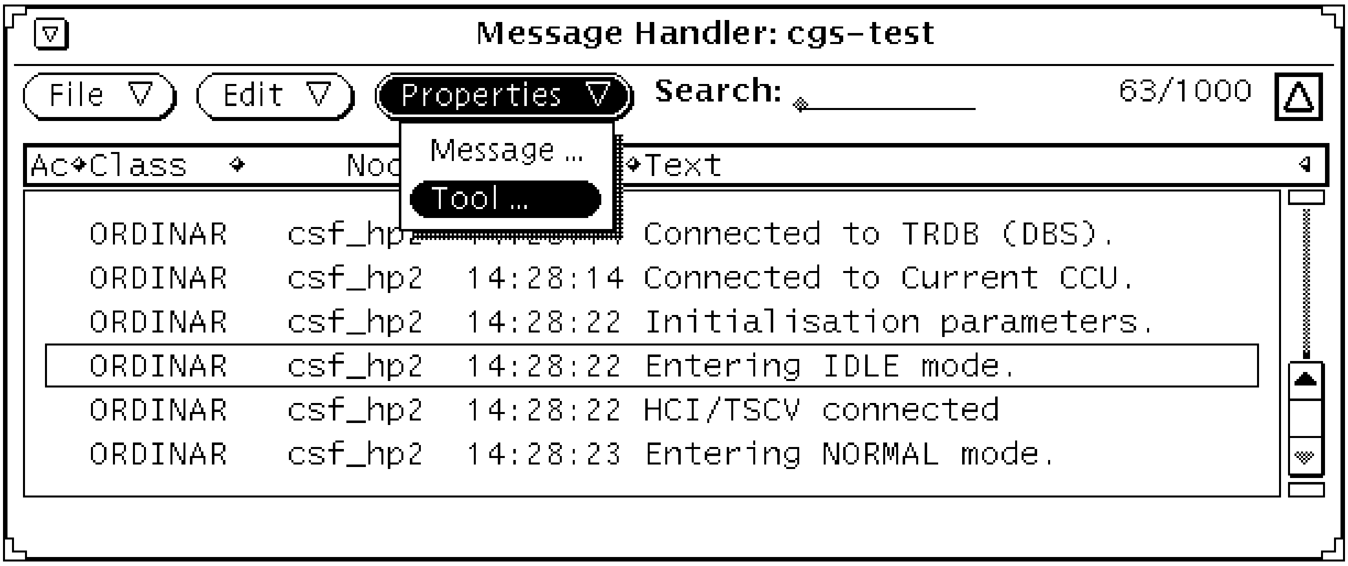


Figure 4–21 : *The Property Menu*

By selecting ”Message ...” the detailed properties of the selected message are displayed. If you select ”Tool ...” the configuration for the tool may be changed, see chapter 4.3.8.

4.3.8 Message Handler Setup

The start up position, size and state (iconic or not) of the Message Handler are configured via command line arguments, just as for any other OpenWindows application, e.g.:

message_handler –Wp 0 0 –Ws 600 200 –Wi

This command starts the Message Handler at position 0/0 (–Wp 0 0), which is the top left corner of the workspace. The main windows size will be 600x200 (–Ws 600 200) and the application will start iconic (–Wi), which means, the main window will be closed and only an icon will be visible at start up.

This line could be entered in a shell or put into your task menu ($HOME/.task_list).

The simplest configuration however is to use the ”Save Workspace” feature of the OpenWindows window manager. Since the Message Handler tends to be started very early in your session this makes sense. You can start it, configure the position and size interactively and then ”save the workspace”. By that, each time you start OpenWindows, which is basic for your session, a the Message Handler is started automatically.

All other settings are configured via a dedicated property window.

To do this, select the **Properties–>Tool** menu entry of the window. A properties sheet (”Message Handler Properties”) comes up which allows to define the layout and contents of the window.

Setup the Message Handler application

- Select **Properties–>Tool...** in the menu line of the **Message Handler** window.

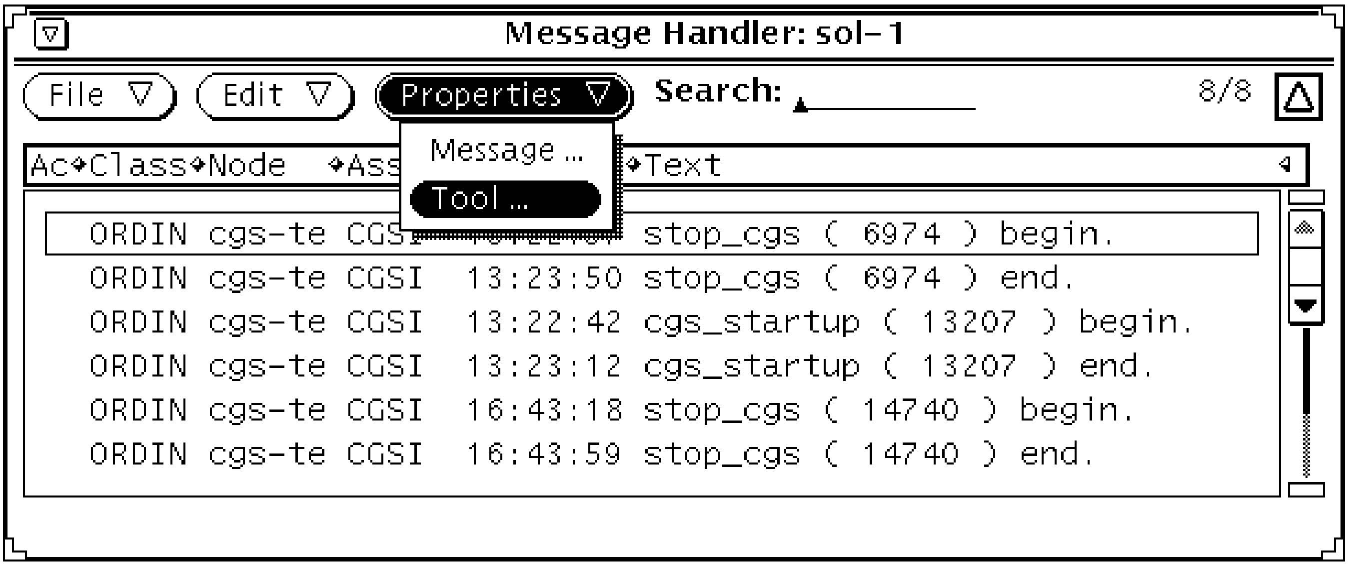


Figure 4–22 : *The Message Handler Properties menu*

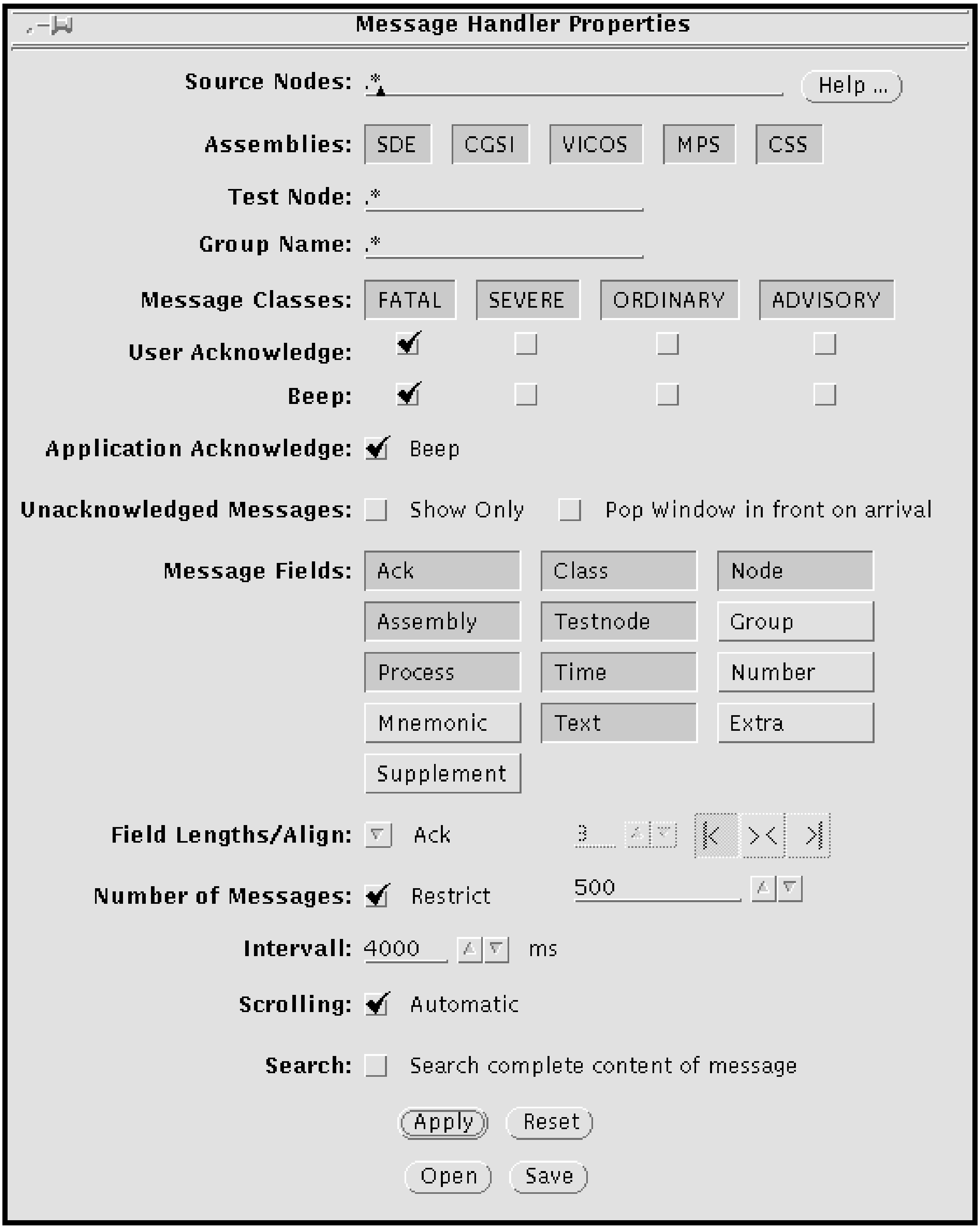


Figure 4–23 : *The Message Handler Properties window*

The Message Handler Properties window contains a panel with the following panel items:

- a **Source Node** specification

  Due to implementation details (RPC broadcasts) and local network configurations (netmasks) it may happen, that your system gets messages from other systems which do not belong to your test environment. Since network topology may not be changed fast enough in all cases, the **Source Node** specification provides a workaround for such a ”noisy” environment.

  You may use a regular expression to specify hostnames, from which you want to receive messages. An online help gives some guidance, push the ”Help”–button and another window pops up:

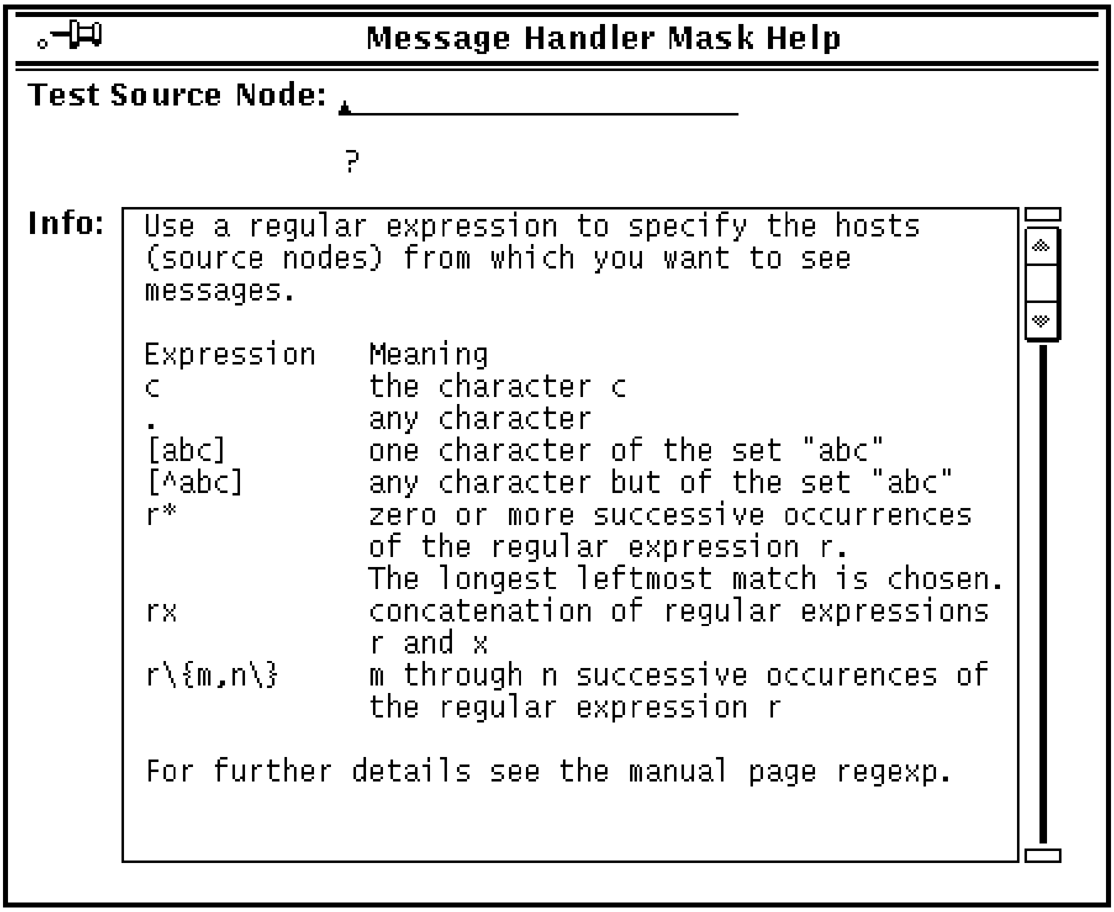


Figure 4–24 : *The Source–Node Help window of the properties window*

In the default situation, the mask is set to ”.*”, which accepts any hostname. you may change it and test it. Just enter a hostname in the help window and hit ”Return”. The question mark below this ”**Test Source Node**” should change to ”Show” or ”Hide” indicating whether messages from that host will be shown or hidden.

Note, that while this is only a workaround for a network configuration problem it may be a worthy filter when evaluating log files.

Note, that this syntax is different from the ones used elsewhere in the properties. For example the ”Test Node” and ”Group Name” use just ”” to specify a wildcard, while the ”Source Node” here is a regular expression yielding the same semantic with ”.*”.*

- a choice list with the title **Assemblies** and the following choices:
  - **SDE**
  - **CGSI**
  - **VICOS**
  - **MPS**
  - **CSS**

  Only messages received from selected assemblies will be displayed. You can see whether a button is selected or not by a look at the button’s appearance. The buttons appear three–dimensional, protruding while not selected. To select an assembly means to press the button which then appears dark with shaded edges. You see an example in Figure 4–25 where the assemblies CGSI, VICOS and MPS have been selected while SDE and CSS are not selected.

- text fields to enter the **Test Node** and **Group Name** that shall be selected for display.

  You may use regular expressions to specify test node and group name. A dot followed by an asterisk (.*) indicates that any name is accepted. These fields are applicable for VICOS only. You have to detect with other means the right names for the entries in this fields. This is described in the VICOS part of this manual.

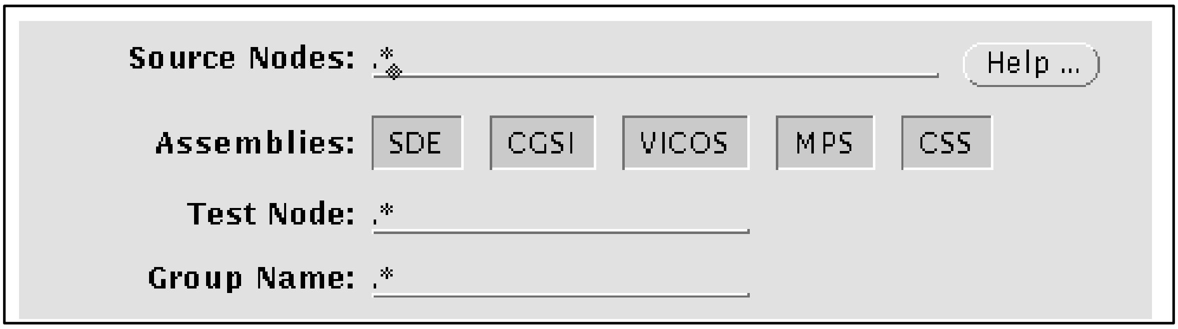


Figure 4–25 : *The Source–Identification fragment of the properties window*

- a choice list with the title **Message Classes**, as depicted in Figure 4–26 below

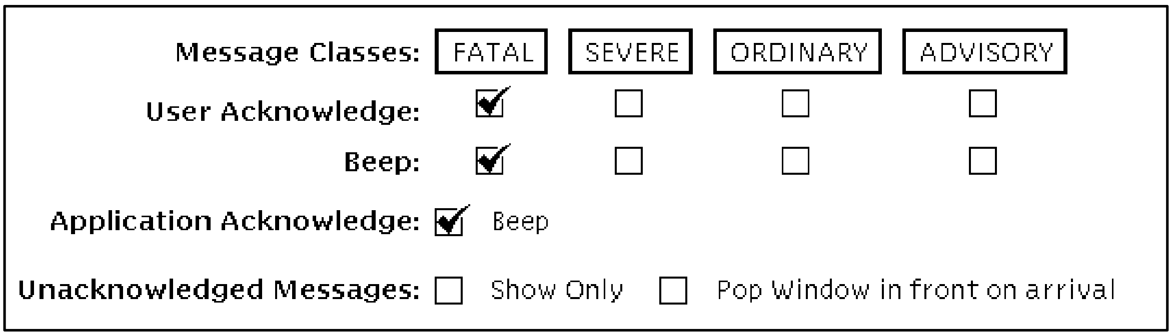


Figure 4–26 : *The Message–Classes fragment of the properties window*

Again the dark and shaded buttons mean that the button is selected. The check–boxes below the message class boxes are only enabled if the corresponding button is selected. The message classes have the following meanings:

- **FATAL**

  A fatal message means indication of an error which is blocking further operation

- **SEVERE**

  A severe message message is indicating a severe error, which affects the execution of all or several processes.

- **ORDINARY**

  Ordinary errors , i.e. errors affecting local processes or disturbing only single operational steps

- **ADVISORY**

  Messages containing information on progress in processing or general advises to the operator. Only messages with selected severity class will be displayed.

• the check list with the title **User Acknowledge**

  Checking the box under the respective class means, that any message received of this class is indicated with an acknowledge indication in the message window.

• the check list with the title **Beep**

  Checking the box under the respective class means, that any message received of this class generates a beep output on your console.

• the check box with the title **Application Acknowledge**

  Checking the box means, that any message with an acknowledge request of the sending application will issue an audio signal.

• the check boxes with the title **Unacknowledged Messages**

  Checking the boxes adjusts the message window to show only messages which have not yet been acknowledged by the operator and if the window shall pop in front on arrival of such messages.

• a choice list with the title **Message Fields** that allows you to define the outline of the message window, as shown in Figure 4–27 below:

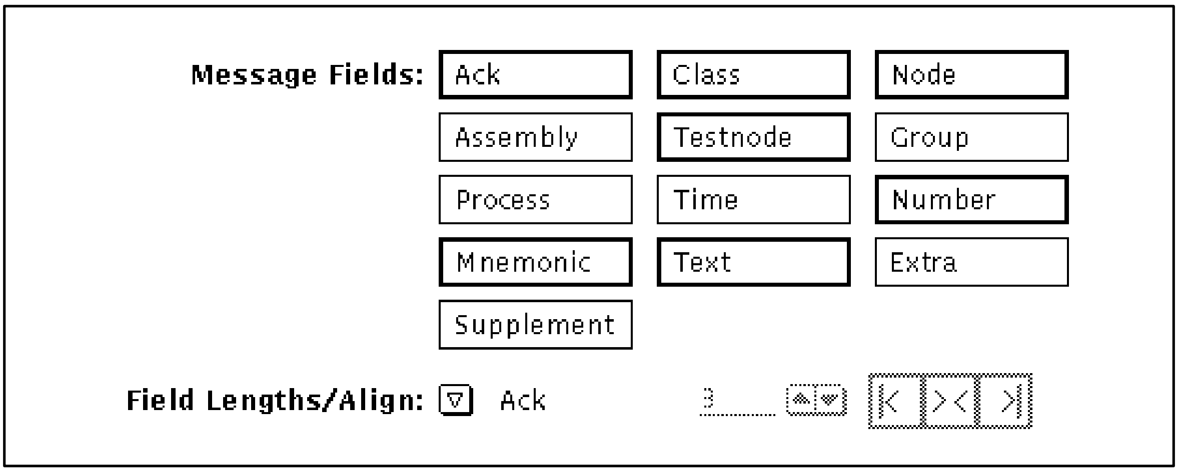


Figure 4–27 : *The Message Fields fragment of the properties window*

A message is composed of the fields indicated. Selecting a field means, that each message in the window will be displayed with this field included. Otherwise the field will not displayed. Again the dark appearance with shaded edges indicates that the button is selected. In the row with the name **Field Lengths/Align** you assign the length in characters for each message field and its alignment. With the small triangle pointing down you select the message field for which you would like to enter the field length. The current setting is visible at the small line to the right. For your convenience there are adjustment support buttons at the right of the line. However, you may enter the length manually by double–clicking into the field and replacing the old value for the length.

Note that some fields have a fixed length that cannot be changed.

Moreover if a field is not selected to be shown, the length can also not be changed. In these cases the Display Lengths field will be greyed out.

The alignment of a field may be of interest for partly hidden information, e.g. if the Time shall show just the time without the date, you might align this field to the right.

The position, size and alignments may be edited in a more intuitive way by directly accessing elements in the header of the list display.

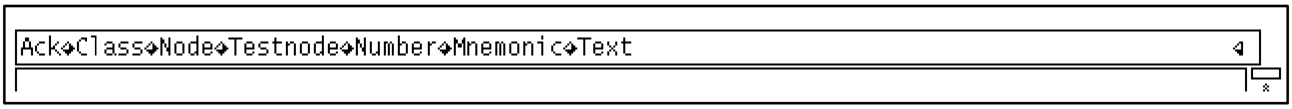


Figure 4–28 : *The Message Fields fragment in the main window*

By dragging the diamond markers it is possible to adjust the size and visibility of fields to the left. By dragging the text title you can change the alignment of this field.

The following fields are indicated:

- **Ack**

  A field in the message that indicates, if the user has to acknowledge the message. The field can be set according to a request given by the generating software, or according to the User Acknowledge settings above.

- **Class**

  The severity class of the message
- **Node**

  The network node (UNIX machine) where the message was generated for.
- **Assembly**

  The assembly having generated the message
- **Test Node**

  The logical name of the test node (for VICOS) where the message was generated (e.g. TES_01, DBS_01)
- **Group**

  The event group for VICOS (for a description see the VICOS part of the manual)
- **Process**

  The number of the UNIX process having generated the message
- **Time**

  The date and time when the message was generated
- **Number**

  The number of the message as given by the generating software, the client process counts its messages starting at 1
- **Mnemonic**

  A short string attached to the message
- **Text**

  The main message text.
- **Extra**

  Additional text given to the message
- **Supplement**

  Additional text given to the message

• some check boxes controlling some special features of the list.

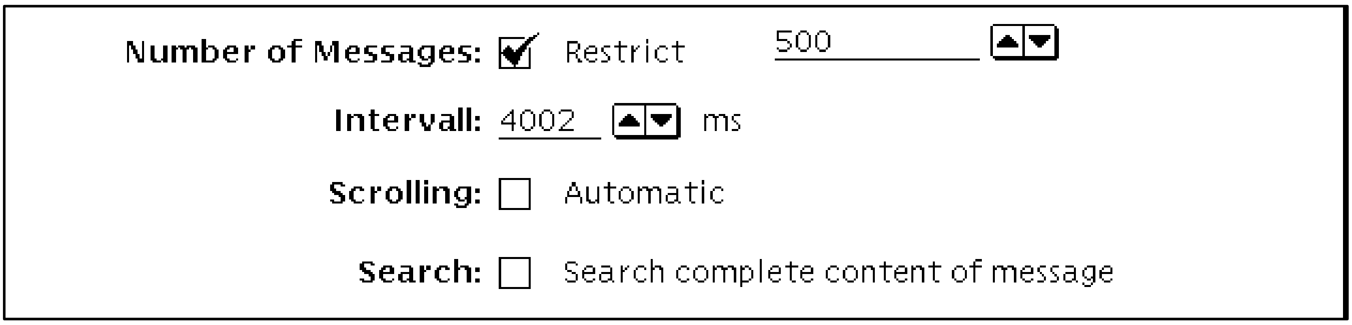


Figure 4–29 : *The List–Internals fragment of the properties window*

- **Number of Messages**

  Activating the check box allows you to define the maximum number of messages held in the message window. For each message there is a record kept in memeory with more than 13 fields and up to 1500 bytes. Performance may decrease if the number of messages grows, so you may try to restrict them to e.g. 500. Note that the field for entering the number is greyed out if the check box is not checked.

Note, that this does not change the log file, which still contains ALL messages and can be viewed at any time by changing this setting.

- **Intervall**

  The list of messages is updated in intervalls, i.e. a timer is set up to clyclically look for new messages. You can see it as a regular change of the frame of the message handler main window. While reading new messages the main window is set to busy, i.e. the frame is greyed out and the cursor, if inside, is set to the clock symbol. If there are no messages, this is just a short flash, if there are more messages, it may take a while. This all varies a bit depending on the rate of messages and the power of your system.
  You can specify the intervall in milliseconds, i.e. if you set this value to 4000 then incoming messages are read all 4 seconds. Note, that larger values increase the overall througput and system performance. Smaller values may make sense if you need a fast response for messages.
  But be careful, setting the intervall to below 1 second may influence other processes too much. It may even hinder interaction with the Message handler itself.
  If you fail to set this value back, you need to kill the Message Handler (consult UNIX manuals for kill or better xkill) and change the tool properties in the file $HOME/.cgsi/message_handler_properties. Look for the line:

```
cgsi.intervall: 4000
```

- **Scrolling**

  If selected, you always see the last received message in the list window. If not selected, the list display is frozen and the operator can search through the window without disturbance by new messages.

- **Search**

  If selected, the search facility not just scans the text displayed in the list, but the complete content of the messages field by field.

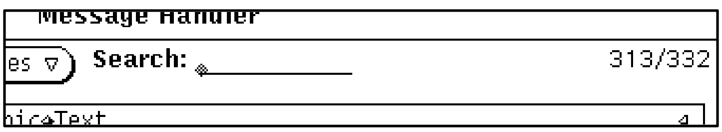


Figure 4–30 : *The Search fragment of the main window*

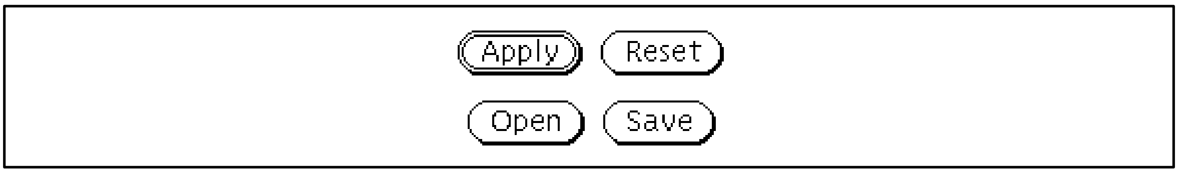


Figure 4–31 : *The Action fragment of the properties window*

- **Apply**

  Activating the button will apply the settings to the message window

- **Reset**

  This button allows you to undo all selections currently made and go back to the selections as in the saved status.

- **Open**

  Activating the button will provide you file a file selector to load new properties. Per default your configuration file is selected.

- **Save**

  Activating the button will provide you a file selector to save the current settings indicated in the Properties Sheet to the file you select. Per default your configuration file is selected. Saving does not include the position and size of the message window. This information can be specified via command line arguments and kept via the OpenWindows window managers ”Save Workspace” menu.

4.3.9 Viewing Messages

To view all fields of a message in detail you need to pop up the message property window.

Select the **Properties–>Message** menu entry of the main window or **double click** on the message in the list.

View the Message Info

- Select **Properties–>Message...** in the menu line of the **Message Handler** window or
- **double click** on a message in the list

Messages that have been determined for acknowledge are indicated with the following symbols in the Ack–Field:

○	The message requires acknowledge and has not yet been acknowledged by the operator
⊠	The message required acknowledge and has been acknowledged.

Figure 4–32 : *The acknowledge indicators*

To acknowledge a message, double–click on a message. The properties sheet for the message comes up. Activate the **Acknowledge** button.

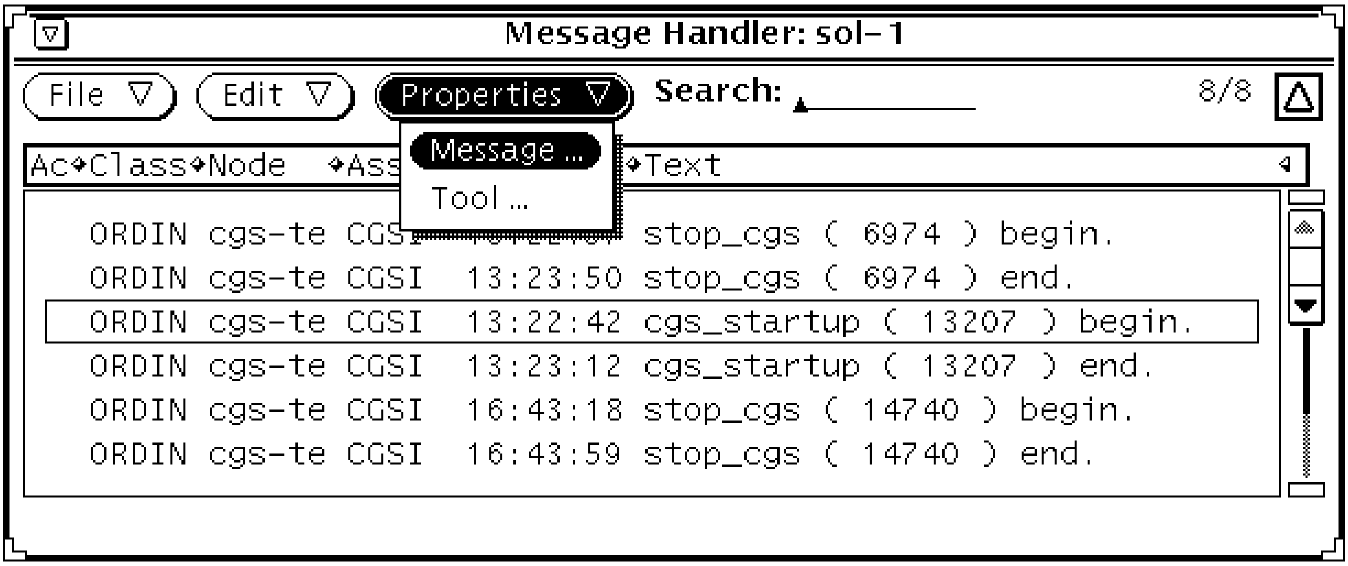


Figure 4–33 : *Open the Message Properties*

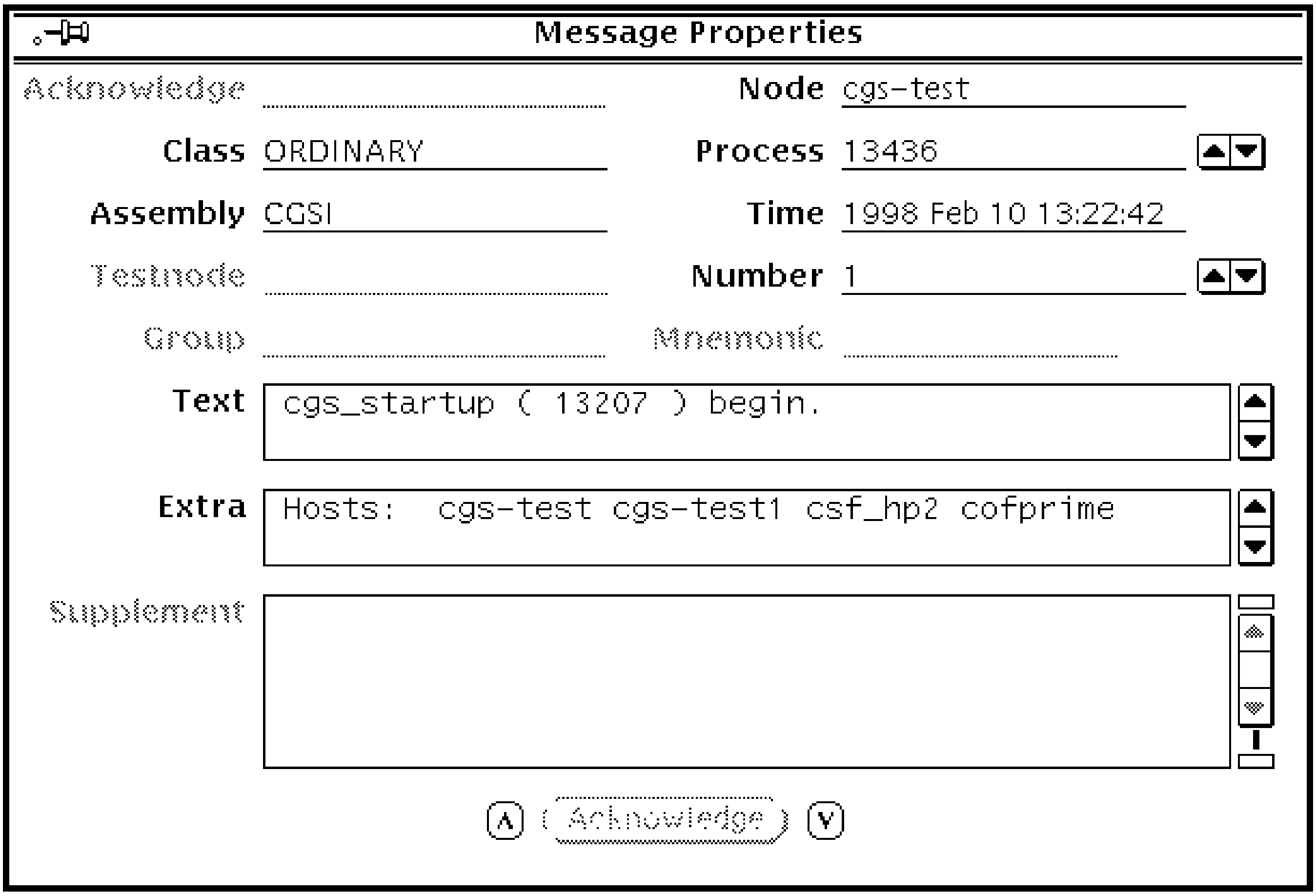


Figure 4–34 : *The Message Properties*

4.3.10 The Icon

To save space in screen area, the Message Handler window may be iconized. There is still an indication for new messages arriving:

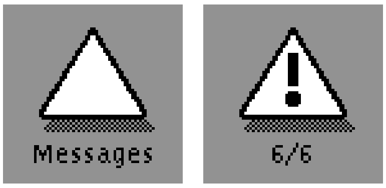


Figure 4–35 : *The Message Handler Icon*

If there are any messages in the list, the number is displayed, just as it is done in the top right corner of the main window. If new messages appear, the numbers are updated. When a message is received, which requires acknowledgement, the icon is blinking red and showing an exclamation mark.

4.3.11 Exiting the Message Handler Window:

To quit the application, select the **Quit** menu entry provided by the window manager, i.e. move the mouse cursor over the top area of the Message handler amin window (the one where ”Message Handler” is displayed), press the menu button (defaults to the right mouse button) and select ”Quit” from the menu popping up.

4.3.12 More about error reporting mechanisms

Beside the CGSI provided message system many tools have their own error reporting system.

The TEV error message reporting approach

TEV distinguishes between two classes of errors: **user input errors** and **software errors** (interface errors with DBS & MPS, file access errors, X errors...).

Software errors are logged in the CGSI Error Message Window and a context–driven message is displayed in the frame footer of the concerned window.

User input errors are logged in the frame footer with the context message. In addition for severe errors a pop–up window will also be displayed to give the user more detailed indication of the error (as long text additional data) and to force the user to notice the message as soon they appear with details to trace the problem.

Note that the message reported in CGSI does not contain the context message of the footer line.

Refer to appendix D-5.1 for more information about the TEV error messages.

The CSS error message reporting approach

CSS distinguishes between two classes of errors: **off–line errors** (errors which occur during model development) and **runtime errors** (error which occur during model execution).

Runtime errors are logged in the CGSI Error Message Window. Additionally they are displayed in the MOCS message window.

Off–line errors are either displayed in the message line of the affected editor window or in a dedicated error window.

Refer to appendix D-2.2.1 for more information about the CSS error message approach.

The MDA error message reporting approach

Some MDA tools (”generate scoe files”, ”consistency checker”, etc.) report to standard output. So be sure to have a console tool open when using MDA tools, since otherwise some important feedback might be lost.

Certainly the CGSI Message Handler should not be used for all kinds of user information. For instance, the interactive tools as the CLS Editor or I_MDB provide most of their information internally (e.g. with pop–up windows). But at present some of these tools create log files or write messages to the console tool.

4.3.13 Error Messages

Refer to appendix D for a detailed description of CGS error messages and how to handle them.

5 DEVELOPMENT SUPPORT SERVICES

5.1 Developing SW in Ada and C

Please note that this section only provides an overview of how to start the SDE tools associated with developing SW in Ada and C. The user should consult the SDE User's Manual documentation listed in section 2.1.1 for more detailed information.

5.1.1 Introduction and Concept

CGS provides a Software Development Environment (SDE) for the development of Ada and C programs. This environment bases on the commercial SDE life*Cycle, which has been adapted to the special needs for CGS. This SDE provides an integrated tool set supporting :

- Requirements Analysis with SADT
- Architectural and Detailed Design in HOOD
- Syntax Sensitive Editing for Ada and C
- Ada Coding by a generic Ada coding environment
- General Coding (in particular for C) by a generic Coding Environment
- Special Pre-compilation of Ada code in terms of translating pathnames to end items in the MDB into short identifiers
- Preparation of documents (e.g. for specifications, design documents, various reports, and test documentation) by an integrated Documentation Facility
- Preparation of Traces by an integrated Traceability and Requirements Management tool

All the tools are integrated and store their results into a central SW development repository. This repository provides also for Configuration Management and version control of the data.

From the users point-of-view he/she will develop Ada and C software using the SDE and storing the results into the Project related SDE data base. This SW can have have

5.1.2 Starting the SW Development Environment

The SW Development Environment can be started from the CGS provided Task Selector, which provides a list of main user tasks. The first task in this list is called Software Development and starts CGS SDE tool. This tool is regarded as a self-standing one, which interfaces to other CGS products via the SW Entity Support tool (SWES). The following procedure gives the instructions on how to access the SDE.

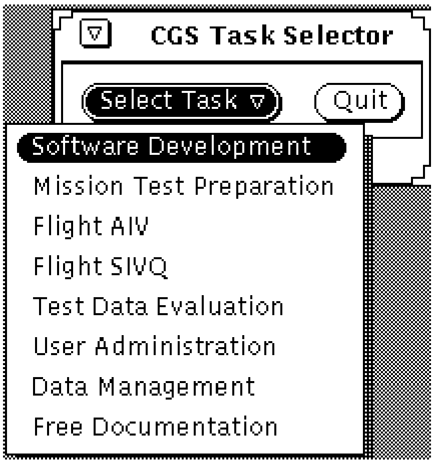


Figure 5–1 : *Starting the CGS SDE*

Starting the SW Development Environment

- **Move** the **mouse cursor** to the menu ***Select Task*** of the ***CGS Task Selector*** window.
- **Hold the right mouse bottom**. You get now a box with all tasks you may select. (ref to Figure 5–1).
- **Move** the **mouse cursor** to the task ***Software Development*** and **release** the **right mouse button**.

The above procedure provides the user with the SDE entry window as shown in Figure 5–2. You can now access one of the projects listed on the bottom part of the window and work with them (see documents listed in section 2.2.1). Only those projects will be listed which you are allowed to use. Also the CM Area can only be accessed, if you are a cm_user.

The database of the projects are stored in the UNIX file system under the user account ”dbadmin” in the directory ”db”. The user dbadmin is the project databases administrator, where the users must be in the same group as user dbadmin.

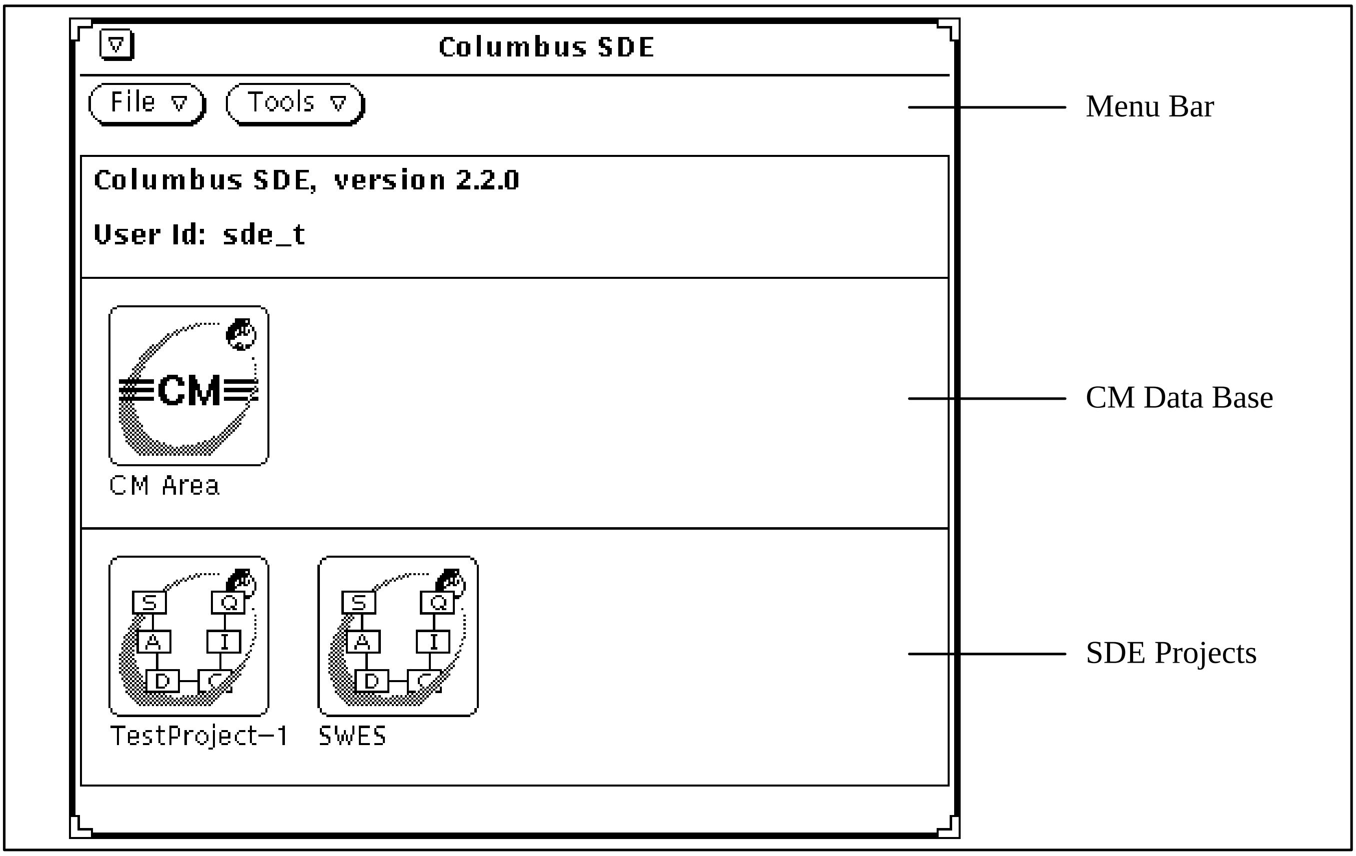


Figure 5–2 : *SDE Main Window (Example from Columbus SDE)*

5.1.3 Standard SDE Functions

Refer e.g. to SDE Manuals (Ref. 2.2.1.1)

5.1.4 Precompilation using the CGS SWES Product within the SDE

To set the MDB environment, the follwing setup is requred:

After having selected the project and the phase CodingUnitTest for it. the selection of SW_Entities and the Properties–>Settings leads to a window, that allows for specification of the DB Environment.
In this Windowentitiled Settings for SW Entities the following items have to be specified:

Object Type:	SW Collection
Display Name:	SW_Entities
Mission_Name:	<Mission to be selected from MDB>
System Tree Version:	<Version Number of MDB System Tree >
CCU Path Name:	<Path of CCU>
CCU Version:	<Version of CCU; Format: Vx.x.x>
CCU Name:	<Name of CCU>

To make the selection effective, select the Apply Button.

To define a Code Unit (Ada) as precompilable, the following is to be done:

Invoke ”Properties–>Settings” for the unit.
A window comes up entitled ”Settings for <unit>”.
Select the checkbox following the ”SWES Precompilable” Line
Make the selection effective by pressing the ”Apply” Button

To precompile a unit with SWES:

Invoke ”Compilation –> SWES Pre–Compile” for the unit.

In case of successful precomilation, the message ”SWES pre–compilation completed” appears on the footer line of the window. In case of errors, appropriate messages are displayed in a message box.

Pre–Compilation Errors can be checked via ”File–>View–>SWES Pre Compilation errors” from the menu. A text editor pops up showing the error messages from the SWES precomilation command execution. The line numbers in the error message file point to the source code lines in source of the unit. The source of the unit can be viewed by ”File–>View–>SWES Pre Compiled Source”

Note:
To define a remote host, where precompilation shall take place, the environment variable

SWES_PRECOMPILER_HOST

must be set to the name of the host. It is necessary, that the filesystem, where the SDE databases reside, must be mounted on the remote host.

5.2 Document Preparation

The CGS document preparation facility is based on the COTS tool Interleaf 5.3 and provides templates for various document types. Please refer to the SDE User's Manuals (see references 2.2.1.1.1 – 2.2.1.1.9 for detailed information).

6 MISSION PREPARATION TASKS

This chapter describes the activities to be performed as part of the mission preparation task.

Based on the overall system requirements specification a system is structured into its elements, sub-systems and further refined components. For all the system constituents the detailed requirements and interfaces have to be described to such a level, that the real production process can be performed on the lowest level, followed by integration and test of the components.

CGS support of the mission preparation task comprises support of the following sub-tasks:

- system and sub-system identification and configuration
- identification of to be developed items (SW and HW)
- identification and definition of necessary data, state codes, etc.
- development and maintenance of SW (Ada, C and APs)
- and Flight Synoptic Displays

Most CGS customers / users have developed hardware components lasting for decades and have a lot of experience with their tools. In contrast, SW development especially in Ada is rather new. Thus it was not the intention of CGS to provide tools for hardware component development; instead the purpose of CGS is to provide tools for mission preparation in terms of mission configuration, component identification and integrated SW development for at least Ada, C, UCL and synoptic displays.

6.1 Mission Configuration

6.1.1 Conceptual Introduction

6.1.1.1 Mission Database Structure

The purpose of the Mission Configuration is to structure the mission into well defined components in terms of system elements, element sub-systems, which again are further broken down to a level identifying items to be developed or used. This break-down is not limited and can be only a few levels, where the lowest level is associated to, for example, an automated procedure, or it can be broken down extensively to identify, for example, a particular screw.

In case of very complex systems, such as space systems, it is obvious, that only the overall mission configuration is performed by the element-contractor – as the prime contractor –, and that the further refinement is performed under responsibility of the sub-contractors involved in building a mission. However, at the end all the bits and peaces, normally already integrated into sub-components, have to be delivered to the element contractor for final integration and check-out of the element. Thus the element contractor must have the complete overview and the overall control of all peaces forming the element.

For this purpose CGS comprises a data base, the so-called **Mission Database**, which allows you to store the complete mission configuration as a hierarchical ’tree’. At the end of the ’branches’ (ie. the ’leaves’), the real data are stored. The ’leaves’ in the CGS world are called **end items**, and are of a particular end item type. All branches between element definitions and end items are of type **virtual node** or **Configuration Data Unit (CDU)**, and are used for structuring purpose.

A node or end item of the mission configuration can be accessed using a so-called **long pathname**, identifying the path in the hierarchy where it can be found. In addition a unique **Short Identifier (SID)** is associated to each node, and is also used for accessing the node.

Now lets have a look on the example in Figure 6–1. The element contractor of the element APM has identified, among others, an onboard thermal control system (TCS) as one sub-system. A sub-contractor has now the task to refine the TCS. For this purpose the **system tree** is frozen by the element contractor and is now available in a certain version. The system tree reflects an element configuration break-down for a particular mission down to a specific level, e.g. down to subsystem or assembly level.

As the next step CGS allows you to create instances of a Mission Database to be distributed to the various sub-contractors. The sub-contractors will now continue with their sub-system configuration based on their MDB instance. Since the system tree is frozen, no sub-contractor will be able to modify this, but instead a sub-contractor can modify his and only his **user-tree**, identified by a **Configuration Data Unit** (CDU). Associated to a user tree is a certain range of SIDs a sub-contractor can use. This range is controlled by the Element-contractor in order to avoid ambiguity of SIDs when the various user trees are later put together.

A CDU comprises virtual nodes for structuring purposes and end items identifying the data being part of it. There are a lot of end item types, which are grouped in so–called **domains**, in order to ensure that only end items of this domain are used by the sub-contractor. For instance the sub-contractor of TCS shall not use end item types of the Electrical Ground Support Equipment (EGSE) within his user tree. Such a domain is defined, when the CDU version is created. Note, that different versions of one CDU can have different domains.

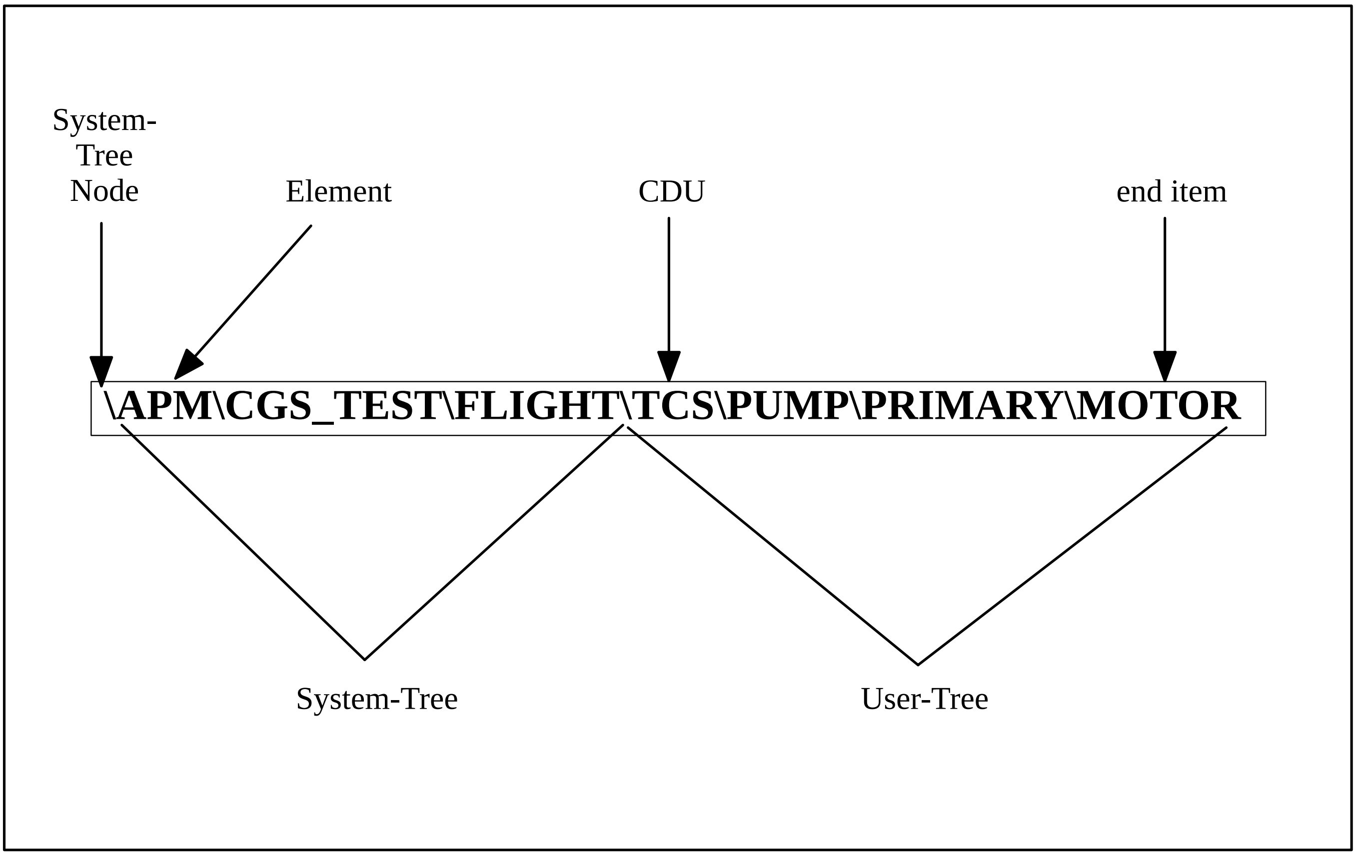


Figure 6–1 : *CGS Pathname Description*

6.1.1.2 Version Control of Mission Configurations

The term CDU version mentioned above leads to the version control concept as used by CGS MDB. This concept allows for version control on three different levels :

<u>System Tree Version Control</u>
During the mission preparation phase it normally happens that system requirements change for various reasons, which often changes the mission configuration as well. However, the element contractor should not just modify the existing system tree, especially if instances have already been created, rather then keeping the actual system tree persistent as an old version.

CGS supports this situation by the creation of a new system tree version, to be modified by the element contractor only. Though, if a certain system tree is referenced, it must be identified by the

System Tree Name, Mission Name, and System Tree Version

*Note, that currently the Mission Name is identified as **Dummy Mission** only.*

<u>CDU Version Control</u>
In contrast to the system tree modification, the CDU change is initiated by requirements, interface, and design changes as well as for bug fixing purposes. In this case it is the sub-contractor who is creating a new version, which is comprised of a leading ’V’ followed by three digits separated by dots (e.g V1.2.3). Usually the version number (first digit) is increased in case of requirements changes, the issue number (second digit) in case of design or interface changes and the revision number (third digit) in case of bug fixing.

CCU Version Control

Configuration Control Units (CCU) are introduced to identify a set of CDUs or even a set of CCUs, which are put together for test, simulation and/or application execution purposes. A CCU can be defined on any level between the element and defined CDUs, where CDUs can be still in the development mode. By this approach the CCU can refer to data as part of a CDU, which are even not yet defined. For this case CGS provides a consistency check facility on CCUs, which allows you to analyse the current status of a CCU.

The version of a CCU is defined the same way as for CDUs, i.e. ’V’, version, issue, revision numbers. Since the definition of a CCU will not change any data, it is allowed for the element contractors as well as for subcontractors.

All of the above three version control mechanisms work only for privileged users having configuration management (cm) right. Only those users can freeze versions, whereas other users can fix a version status only by setting a version to ’for review’.

For a detailed description of the Mission Database concepts, please refer to document [2.1.1.2].

6.1.2 How to Build a Mission Configuration

This section describes how to build mission configuration by creating Elements, CDUs, virtual nodes, CCUs and end items. For detailed information please refer to [2.1.1.3]. Note, that the term ’element configuration tree’ refers to both, the system tree and the user trees of a corresponding element

6.1.2.1 Starting a Mission Configuration Session

The first action to do is to invoke the appropriate CGS tool supporting the mission configuration phase. This tool is known as I_MDB (Interactive Mission Database access) and will provide the user with a window called ’I_MDB Navigator’.

Starting a Mission Configuration Session

- **Press** the ***Select Task*** button of the ***CGS Task Selector*** window with the right mouse button.
- **Hold the right mouse bottom**. You get now a box with all tasks you may select. (ref to Figure 6–2).
- **Move** the **mouse cursor** to the task ***”Mission Preparation”*** and **release** the **right mouse button**.

Note, the contents of the task list is not fixed and can be modified for each user by editing the ’.task_list’ file. Figure 6–2 shows the standard CGS Task Selector

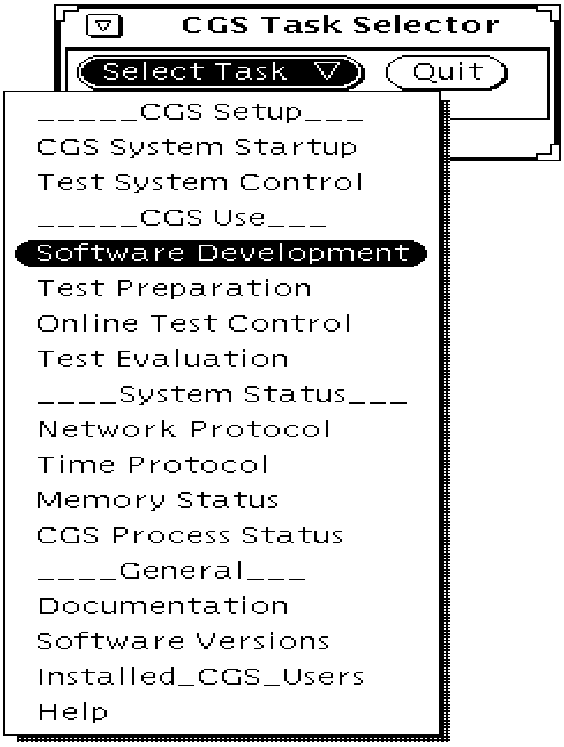


Figure 6–2 : *Mission Configuration Start from CGS Task Selector*

The ’I_MDB Navigator’ window is shown in Figure 6–3. From here you can model and navigate through element configuration trees by using the provided direct interactive access to the Mission Data Base.

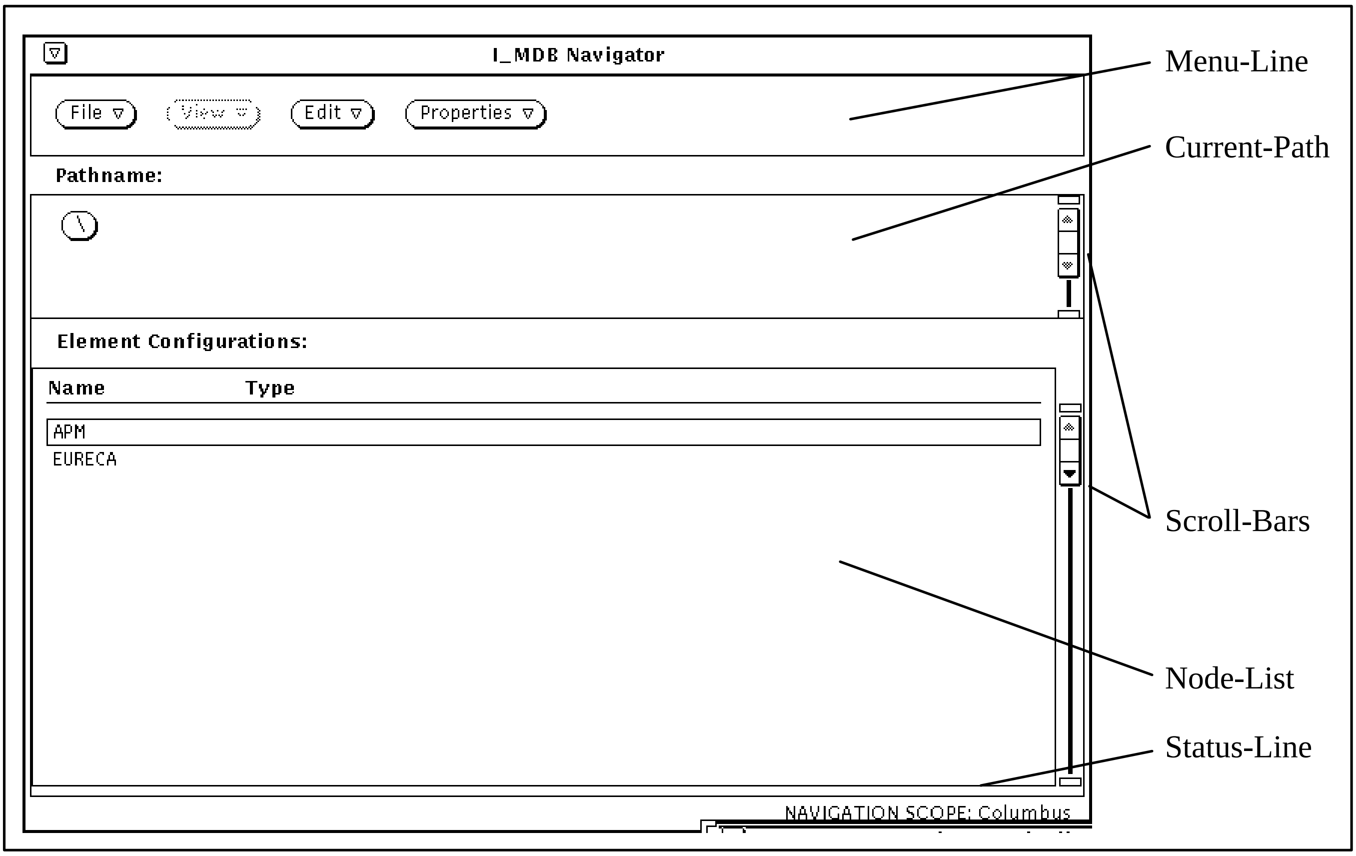


Figure 6–3 : *The I_MDB Navigator Window*

The I_MDB Navigator window is divided into several parts :

- On the top of the window the Menu-Line displays menu buttons with context sensitive functions associated. The usage is partly explained in this chapter and fully described in document [2.1.1.3].
- In the middle part, the Current-Path provides the actual path to the node the user has navigated to. After first invocation the system tree node '\' is displayed and when the user navigates through the tree, the Current-Path is updated depending on stepping down or up in the element configuration tree.
- Below the Current-Path, the Node-List provides the user with a list of currently accessible nodes being forming part of a mission configuration. Depending on the previously selected node, one of the following categories are listed :
  - all Element Configurations, if the current path is the system tree node
  - System Tree Nodes, if the current path is within a system tree of a selected element configuration
  - User Tree Nodes, if the current path is within a selected CDU
- The Status-Line provides the user with various messages.

Note, that the current version of CGS does not provide features for creation and deletion of elements. Thus you will have pre-defined elements listed under Element Configurations.

The functions associated to the buttons of the Menu-Line are disabled until a selection has been made; except for the **Properties–>Tool...** button, which lists information on the tool and user. This is not explained here in detail.

The Scroll-Bars at the right site of the Current-Path and Node-List can be used in case that the provided information can not all be listed on the screen, e.g. if navigation has been performed rather deeply in the hierarchy or more than 13 nodes are listed. The usage of the Scroll-Bars is as follows:

Using Scroll-Bars in I_MDB Navigator

- **Move** the mouse to the **Scroll-Bar arrows** on the right site of the window. The arrow pointing down means scrolling down where the arrow pointing up means scrolling up.
- **Click** with the **left button** on the **arrows**. This provides you with one more line you can select from.
- **Hold** the **left button** to scroll fast up or down. This provides you with some more lines depending on how long the button is hold down.

6.1.2.2 Navigation within Element Configuration Trees

First we will explain how to navigate through the overall element configuration tree. This is necessary in order to work later with the system/user tree nodes or end items. For the following procedure it is assumed, that the I_MDB Navigator window has opened with the system tree node as Current-Path.

Navigating down within a System Tree

Note, that the use of the Scroll-Bars may be necessary, but is not explicitly mentioned.

- **Double click** on the element you want to navigate to. This results to a box called ***System Tree Version*** listing all available version of the element system tree and the corresponding CM status.

The double click does only work on non-selected nodes (on selected nodes frame borders are visible). Otherwise the following optional step has to be performed.

- **Hold** the **right mouse button**, a menu pops-up, **move** the **mouse cursor** to the entry ***Show System Tree Versions ...*** and **release** the **mouse button**. The **System Tree Version** box pops up.
- **Double click** on the version you want to navigate to. Now the I_MDB Navigator window is updated as follows :
  - The Current-Path shows the Element Name and its system tree version.
  - The Node-List provides a list of all system tree nodes forming part of this version
- if a version is already selected press the **Command button** with the right mouse button and select **Command–>Open and Dismiss Window**.
- **Double click** on the system tree node. This opens the next lower level of the selected system tree node, if this is a virtual one, or it opens a CDU (see below). The I_MDB Navigator window is updated w.r.t. Current-Path and Node-List.
- or (if already selected) **hold** the **right mouse button**, a menu pops-up, **move** the **mouse cursor** to the entry ***Open*** and **release** the **mouse button**.

The above procedure works recursively for any hierarchy level within the system tree of an Element Configuration unless the selected node is of type CDU. The following procedure will now describe how to access a CDU version and the navigation within a CDU.

For the procedure below it is assumed that a system tree version of an Element Configuration has be accessed and that the system tree node list contains the CDU to selected.

Navigating down within a User Tree Version

- *Note, that the use of the Scroll-Bars may be necessary, but is not explicitly mentioned.*
- **Double click** on the to be selected CDU. This opens a box called ***I_MDB: CDU Versions*** (see Figure 6–4) listing all the versions of the selected CDU and their status.
- *Also here the double click does only work on non-selected nodes. On already selected nodes (frame borders are visible) the following optional step has to be performed.*
- **Hold** the **right mouse button**, a menu pops-up, **move** the **mouse cursor** to the entry ***Open*** and **release** the **mouse button**.
- **Double click** on the **CDU version** you want to work with. This action leads back to the I_MDB Navigator window which now lists :
  - The path to the CDU version as the Current-Path
  - A list of user tree nodes forming part of the CDU version in the Node-List
- or (if a CDU version is already selected) **press** the **Command** button and **select** the **Command–>Open and Dismiss Window** option.
- **Double click** on the user tree node. This opens the next lower level of the selected CDU and an updated I_MDB Navigator window w.r.t. Current-Path and Node-List.
- *Again, the double click does only work on non-selected nodes (on selected nodes frame borders are visible). Otherwise the following optional step has to be performed.*
- **Hold** the **right mouse button**, a menu pops-up, **move** the **mouse cursor** to the entry ***Open*** and **release** the **mouse button**.

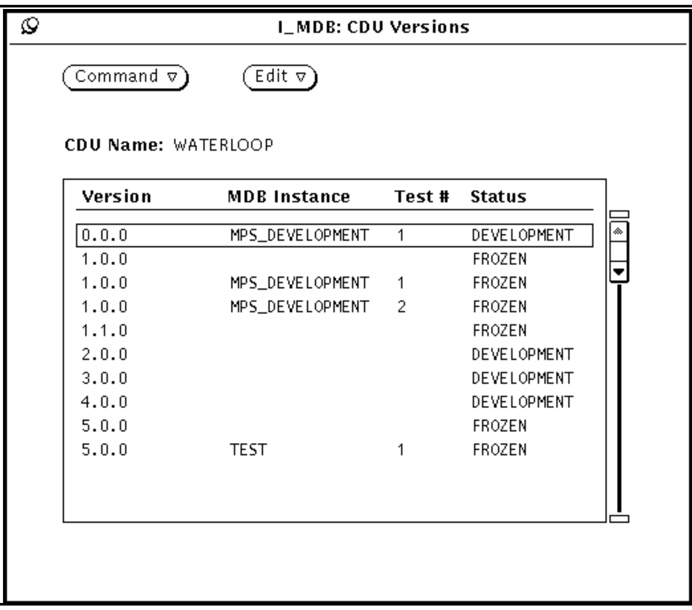


Figure 6–4 : *I_MDB: CDU Versions Box*

The ’I_MDB: CDU Versions’ box in Figure 6–4 lists all the versions of a CDU including CDU test versions. CDU test versions are created in case of changes shall be performed on the user tree by organisation who do not have access to the appropriate MDB instance. If a test CDU is created, this does not belong to the original MDB instance. Refer to document [2.1.1.2] for more detailed information.

A special way for navigation represents the selection of a Configuration Control Unit (CCU) within the system tree, but beneath an Element. By selecting a CCU version a special view on the element configuration tree is created, providing only those CCUs and CDUs defined for the selected CCU version.

A CCU can be selected either from a complete list of all CCU versions defined for the selected element or from a list of all CCU versions defined for a particular System Tree node :

Note that a virtual System Tree Node must be selected in the Current_path field.

Navigating down to a CCU version from the complete list

- **Move the mouse** to the Menu-Line and **select File–>Browse All CCU Versions...** . This opens the ***Browse CCU Versions*** box listing all defined CCUs including pathname, CCU Name and CCU version.
- **Use the Scroll-Bar** on the right site of ***Browse CCU Versions***. (works like for the I_MDB Navigator window, see above)
- **Click** on the CCU you want to access and **click** on the ***Apply*** button.

The I_MDB Navigator window is now updated in two parts :

- The Current-Path shows now the path nodes down to the CCU, where at the end of the path the name and the version of the CCU is displayed.
- The Node-List does now contain only
  - the CDUs
  - and the virtual system tree nodes reflecting CCUs

  forming part of the selected CCU version.

The above described update of the I_MDB Navigator window does also apply for the second way of navigating to a particular CCU version.

Navigating to a CCU of a particular System Tree Node

- **Click** on the last entry of the Current-Path (must be a virtual system tree node)
  or
  click on a virtual system tree node within the Node-List of the I_MDB Navigator window.
- **Move the mouse** to the Menu-Line and **select** ***File–>Show CCU Versions...*** . This opens the ***I_MDB: CCU Versions*** box listing all defined CCUs which root is the selected system tree node (see example in Figure 6–5) .

  The ***I_MDB: CCU Versions*** box is divided into two parts. The left part lists the names of all CCU names being defined. By selecting one CCU, the right part is updated, listing all versions and their status created for the selected CCU.
- **Click** on a CCU in ***CCU Names*** . The right part of the ***I_MDB: CCU Versions*** box is updated with version informations.
- **Double click** on a CCU version.

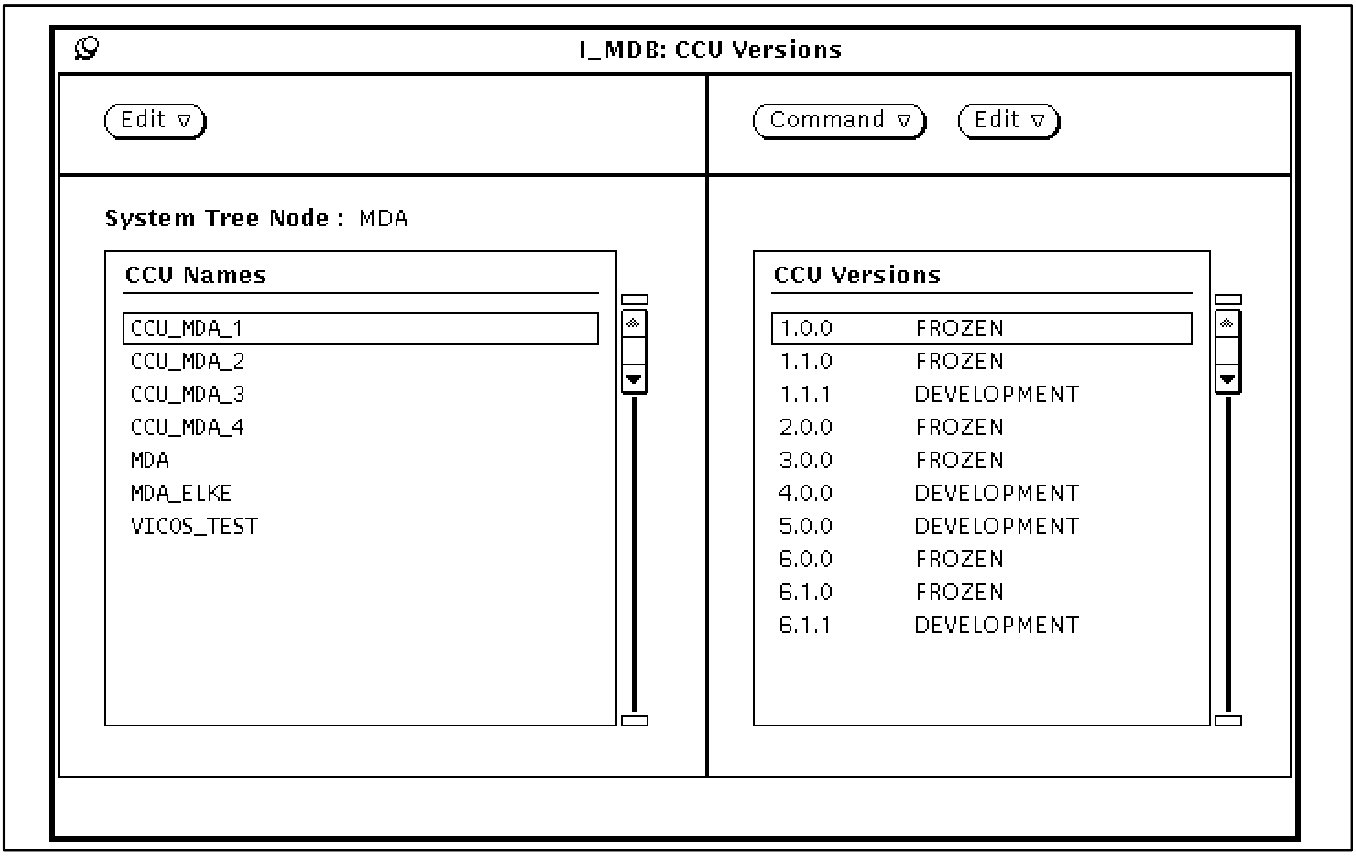


Figure 6–5 : *I_MDB: CCU Versions*

Note, that once a CCU has been selected you will not be able to select another CCU version within the current one. For this purpose you have to navigate upwards in the element configuration tree (see below)

Up to here it was described how to move down in the element configuration tree. Moving upwards is much easier and works always the same.

Navigating upwards in the Element Configuration Tree

- **Move** the mouse into the Current-Path part of the I_MDB Navigator window.
- **Double click** on the path-node you want to move to. This updates the I_MDB Navigator window with the new node informations associated to the selected path-node.

Note, if a path-node higher than a selected CCU is selected, the current CCU version is no longer applicable.

6.1.2.3 Creating Nodes in a Element Configuration Tree

Since the element configuration tree as defined above being a system tree plus the corresponding user trees, the creation of nodes is foreseen in terms of

- Creation of Elements
- Creation of System Tree Nodes
- Creation of User Trees (CDU)
- Creation of User Tree Nodes
- Definition of Configuration Control Units (CCU)

However, the current version has not reached its final implementation, and therefore the following note has to be given.

The current CGS Version will provide predefined System Tree Versions which are FROZEN so that they cannot be updated or deleted in any way. Thus it is not possible to create Elements or virtual System Tree Nodes.

The creation and modification of nodes within the element configuration tree shall not be performed by any user, but only by personal with MDB CM access. This has to be checked first.

Checking CM Status

- **Move** the mouse to the Menu-Line.
- **Select Properties–>Tool**. This provides you with the tool and user properties.
- **Check** that the **CM Status** is set to **Yes**. If this is not the case, you will not be able to perform the actions described in the following.
- **Close** the ***Tool Properties*** box by **clicking** on the **pin** in the upper left corner.

The first node to be created is a CDU, which than contains user tree nodes.

Creating initial Configuration Data Unit (CDU)

- **Navigate** to the system tree node where a new CDU shall be placed.
- **Move** the mouse to the Menu-Line and **select File–>Create CDU...**. This will provide you with the ***Create CDU*** box.
- **Enter** the *<cdu–name>* and **click** on ***Apply.*** Now the new CDU is placed in the list of System Tree Nodes.

Now with the above procedure only an initial CDU has been created, but a CDU version has not been defined yet. This is done by the following procedure which works also for CDU where versions are already defined.

Creating a new CDU Version

- **Double click** on the newly created CDU. This opens an empty ***I_MDB:CDU Versions*** box similar to Figure 6–4 but without any contents.
- **Select** ***Edit–>Create...*** from the menu buttons on the top of the box. This opens the ***Create CDU Version*** box. A detailed description is provided below.
- **Click** on one of the 4 options to specify a new version in the ***Create CDU Version*** box. The selection will be highlighted by a bold frame.
- If the new version shall be a copy from an old one, click on the ***Copy from CDU Versions...*** button. This provides you with the ***I_MDB: CDU Version list*** box. This looks similar to Figure 6–4, except that there is an ***Apply*** button at the bottom.
- **Click** on a version listed in the ***I_MDB: CDU Version list*** box and **click** on ***Apply.*** The version number, incl. the test CDU version reference (if applicable) is displayed behind the ***Copy from CDU Versions...*** button.
- **Click** on the **CDU domain...** button to define the domain. This provides you with the ***CDU domain list help*** box, listing all possible CDU domains.
- In the ***CDU domain Help list*** box, **click** on the CDU domain to be selected.
- **Click** on the ***Apply*** button. Now the name of the CDU domain is displayed behind the ***CDU domain...*** button in the ***Create CDU Version*** box.

The definition of a CDU domain must not be performed in case of copy a CDU version.

- **Click** on the ***Owner...*** button. This will provide the ***User List*** box, listing all known user of the MDB instance.
- In the ***User List*** box, **click** on the user to be selected as owner.
- **Click** on the ***Apply*** button. Now the user name is displayed behind the ***Owner...*** button in the ***Create CDU Version*** box.
- **Enter** some <*description test*> behind the field ***Description :***.
- **Click** on the ***Apply*** button of the ***Create CDU Version*** box. Now the latter box disappears and the new CDU version is listed as part of the ***I_MDB: CDU Version*** box.

The ***Create CDU Version*** box (Figure 6–6) contains several items:

- In the first row you have to decide if you want to create a new version, issue, revision or a new test version. If a new CDU is created, a new version must be created in any case !.
  If a new issue or revision shall be created for an existing version, the latter must have the status **FROZEN**.
- The ***CDU domain*** is a mandatory field, which definition defines the set of end items which can be created within this CDU version. The domain is decided automatically in case that a copy from CDU versions has been performed.
- The ***Copy from CDU Versions*** must be performed in case that a new issue or revision shall be created. In other cases the can be filled optionally.
- The ***Owner*** is also a mandatory field to decide the owner of a CDU version. The owners of different versions of a CDU may differ.
- The field ***Description*** can optionally be used for comments to a CDU version.

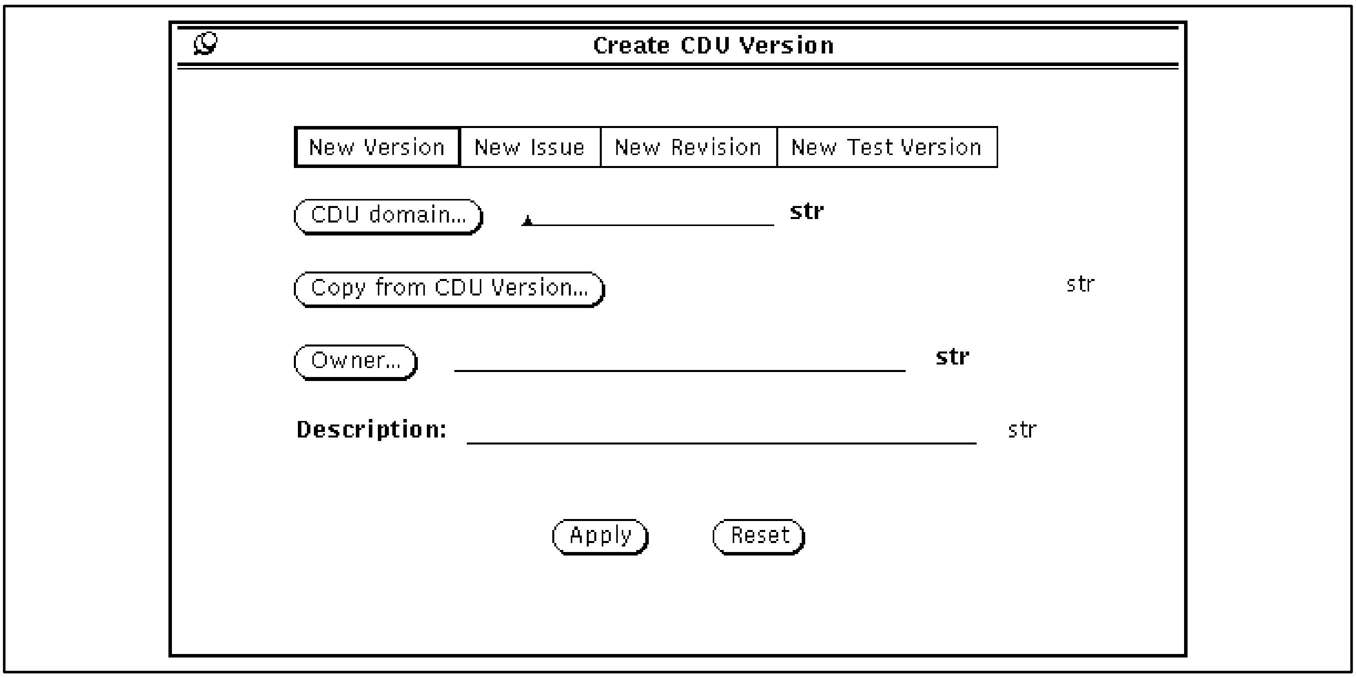


Figure 6–6 : *Create CDU Version Box*

Having accessed a CDU version, new User Tree Nodes can be created within this user tree. User Tree Nodes are either virtual or end items. The user tree node type 'virtual' is the default type. End item types have to be selected from a list of available end items depending on the CDU domain. As part of this section only the creation of the virtual user tree nodes are described, because those are used to structure the overall mission configuration. The creation of End Items is explained in section 6.2.

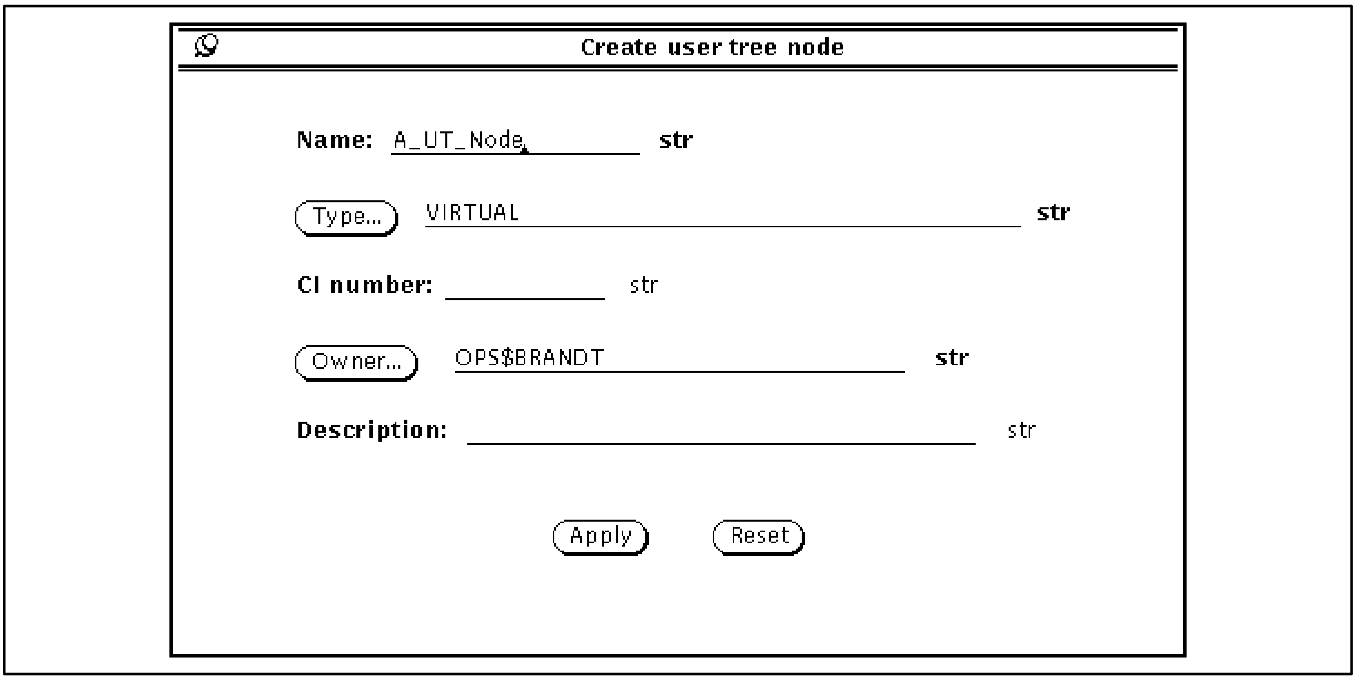


Figure 6–7 : *Create user tree node box*

Creating User Trees Nodes

- **Navigate** into a particular CDU version.
- **Select File–>Create Node...** in the command part of the I_MDB Navigator window. This opens the ***Create user tree node*** box (see Figure 6–7).
- **Enter** the <*user–tree–node–name*> behind the entry **Name.**
- **Click** on the ***Type...*** button. This will provide the ***Node type list help*** box, listing all possible node types.
- In the ***Node type list help*** box, **click** on the desired node type, then **click** on the ***Apply*** button. Now the node user nametype is displayed behind the ***Type...*** button in the ***Create user tree node*** box.
- **Enter** a Configuration Item number, if applicable. (Configuration Items (CI) are used to identify system components formally to be delivered.
- **Click** on the ***Owner...*** button. This will provide the ***User List*** box, listing all known user of the MDB instance.
- In the ***User List*** box, **click** on the user to be selected as owner, then **click** on the ***Apply*** button. Now the user name is displayed behind the ***Owner...*** button in the ***Create user tree node*** box.
- **Enter** some <*description test*> behind the field ***Description :***.
- **Click** on the ***Apply*** button of the ***Create user tree node*** box. Now the latter box disappears and the new user tree node is listed as part of the Node-List of the I_MDB Navigator window

All the creation of CDUs and user tree nodes is mainly done to structure the lower level mission configuration. For the purpose of application execution and test it is necessary to select a set of CDUs and even CCUs versions to create a new CCU version.

Create a new Configuration Control Unit (CCU)

Note that virtual tree nodes can be the root of a CCU only.

- **Navigate** to the path-node which shall be the root of a CCU.
- **Click** on the last entry of the Current-Path (must be a virtual system tree node)
  or
  click on a virtual system tree node within the Node-List of the I_MDB Navigator window.
- **Move the mouse** to the Menu-Line and **select File–>Show CCU Versions...** . This opens the ***I_MDB: CCU Versions*** box (see Figure 6–5) listing <u>all</u> defined CCUs which root is the selected system tree node.
- **Select Edit–>Create...** on the upper left site of the ***I_MDB: CCU Versions*** box. This provides the ***Create CCU*** box.
- **Enter** the <*new–ccu–name*> in the ***Create CCU*** box and **click** on the ***Apply*** button. Now the new CCU is listed on the left part of the ***I_MDB: CCU Versions*** box.

Create a CCU Version

- *Note that virtual tree nodes can be the root of a CCU only.*
- **Navigate** to the path-node which shall be the root of a CCU.
- **Click** on the last entry of the Current-Path (must be a virtual system tree node)
  or
  click on a virtual system tree node within the Node-List of the I_MDB Navigator window.
- **Move the mouse** to the Menu-Line and **select File–>Show CCU Versions...** . This opens the ***I_MDB: CCU Versions*** box (see Figure 6–5) listing all defined CCUs which root is the selected system tree node.
- **Click** on one of the listed CCU names at the left part of the **I_MDB: CCU Versions** box. This will list all available CCU versions of the selected CCU on the right side.
- **Select Edit–>Create...** on the upper right site of the ***I_MDB: CCU Versions*** box. This provides the ***Create CCU Version*** box (see Figure 6–8).
- **Click** on the **New Version** option to specify a new version in the ***Create CCU Version*** box. The selection will be highlighted by a bold frame.
- *To create a new issue or a new revision* ***click*** *in the preceding step either the button* ***New Issue*** *or the button* ***New Revision****.*
- **Click** on the ***Copy from CCU Versions*** button (this is optionally if a new version shall be created). This opens the ***CCU Version list*** box listing all available CCU versions for the selected .
- **Click** on a CCU version in the ***CCU Version list*** box and **click** the ***Apply*** button. This copied version number is displayed behind the ***Copy from CCU Versions*** button.
- **Click** on the ***Owner...*** button. This will provide the ***User List*** box, listing all known user of the MDB instance.
- In the ***User List*** box, **click** on the user to be selected as owner followed by a **click** on the ***Apply*** button. Now the user name is displayed behind the ***Owner...*** button in the ***Create CCU Version*** box.
- **Enter** some <*description test*> behind the field ***Description :***.
- **Click** on the ***Apply*** button of the ***Create CCU Version*** box. Now the latter box disappears and the new CCU Version is listed as part of the right site on the ***I_MDB: CCU Versions*** box.

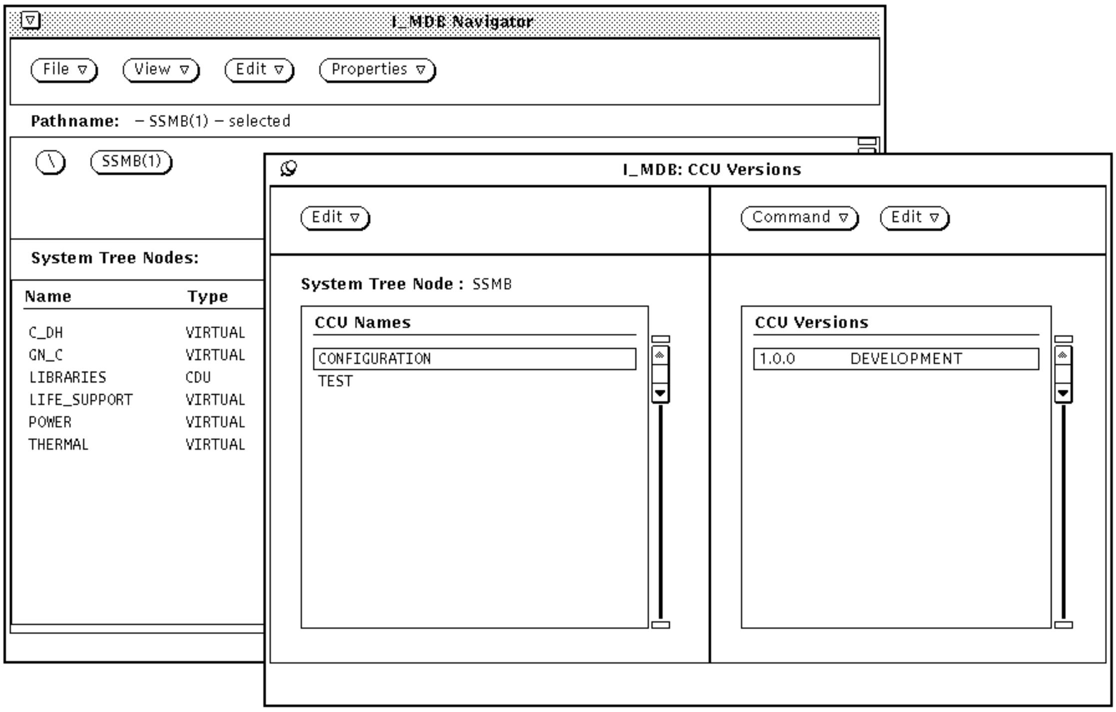


Figure 6–8 : *Create CCU Version Box*

Now a new version of a CCU is created, but the contents is not necessarily defined. This is especially true if a completely new CCU has been created or the CCU version has not been copied from an existing ones, because then the CCU version is empty. In the following it is described how the contents of the CCU is specified.

Specifying CCU Version Contents

- **Navigate** to the path-node which shall be the root of a CCU.
- **Click** on the last entry of the Current-Path (must be a virtual system tree node)
  or
  click on a virtual system tree node within the Node-List of the I_MDB Navigator window.
- **Move the mouse** to the Menu-Line and **select File–>Show CCU Versions...** . This opens the ***I_MDB: CCU Versions*** box (see Figure 6–5) listing all defined CCUs which root is the selected system tree node.
- **Click** on one of the listed CCU names at the left part of the ***I_MDB: CCU Versions*** box. This will list all available CCU versions of the selected CCU on the right site.
- **Click** on one of the listed CCU versions at the right part of the ***I_MDB: CCU Versions*** box and **select Command–>Maintain References...** from the menu buttons on top of the box. This opens the ***I_MDB Maintain References*** box.

  The ***I_MDB Maintain References*** box contains two parts. The upper part lists all referenced CCU versions, where on the bottom all CDU references are listed. The following steps are exactly the same for the creation of both kind of references.
- **Click** on the ***Edit*** button for CCUs/CDUs. By this the ***Browse CCU/CDU Version*** is displayed, listing all CDU/CCU created beneath this system tree node.
- **Click** on a CCU/CDU you want to refer and **click** on the ***Apply*** button. Now the referenced CDU/CCU is listed in the appropriate part of the ***I_MDB Maintain References*** box.
- When all references have been established, **click** on the **Pin** on the upper left corner of the ***I_MDB Maintain References*** box
- **Select** the **Command–>Open and Dismiss Window** to navigate to the maintained CCU version.

*The selected CCU is displayed in **bold** type in the Current_path part of the I_MDB navigator.*

This finalizes the section on how to create user trees, user tree nodes and CCUs. For more detailed information please refer to the corresponding detailed user manual [2.1.1.3].

6.2 Creation and Contents Definition of Data

6.2.1 Introduction

The definition of data is performed on an Element Configuration by creating end items as part of the user trees (CDU), that are also stored in the Mission Database (MDB) . End items represent the lowest level individual reconfigurable items like automated procedures or commands and are modelled by the leaf nodes of the user tree.

In contrast to virtual items (refer to section 6.1.1), end items contain all detail information describing specific end item characteristics. Thereby end items can be comprised to different end item classes (end item types) where all end items of a class (type) are characterized by the same set of attributes.

Example1:
All automated procedures are described by the attributes parameter list, source code, etc.. A specific automated procedure 'Activate Pump of the APM TCS' will be stored in the MDB together with its concrete parameter list, source code, etc..

Example2:
If the same error message information is used for several limit sets, it is unsatisfying for a user to be forced to enter exactly the same error message data again and again for each of these limit sets. In this case it is rather desirable that the user once defines an end item of an MDB provided end item type 'TCS Error Message' which can then be referenced by all corresponding limit sets.

Especially Example 2 shows that the constitution of end item classes and the ability to reference end items are extremely useful in avoiding redundancies.

A lot of end item classes are predefined in the MDB together with the attributes which characterize each class. In addition CGS provides for the possibility to create new end item classes, but it is important that those should base on the existing classes. This is especially important if those end items shall later be used within an Engineering Ground Support Equipment (EGSE) configuration of CGS to be used for final check-out.

Note, if a new end item class does not base on an existing one, a small utility has to be developed by at CGS customer/user site to map the new end item class to existing end item classes.

For this overall CGS User Manual the customization of end items is not described further. Please refer to document [2.1.1.3] for more information on this subject.

From the CGS users point of view the definition of end items is done in different procedural behaviour, which can be grouped as follows :

1. End items containing purely a set of data. For those end items various masks are provided where the user can enter the relevant data. This is for most of the end items.

2. End items invoking special tools necessary to create the relevant data. These end items are those for the definition of automated procedures, simulation models and synoptic displays.

3. End items referring to Software Replaceable or Software Exchangeable Units (SWRU, SWEU). In this case parts of a data set are down-loaded from the SDE repository into the MDB.

In the following the various procedures are described in separate sub-sections.

We should like to put your attention to the appendix 7.6.2, where most end items are described.

6.2.2 How to Define End Items Containing Data

As already mentioned above data definitions are represented as end items in the MDB, which are placed beneath a specific user tree, i.e. in a CDU version. Thus before an end item shall be created you have to navigate to the path-node of a user tree you are allowed to work with. Here you can now create a new node and select the specific end item type (different of type ’virtual’).

Caused by the fact that their are quite a lot end item types and because the user can define their own end item types, the creation of an end item is only performed exemplary, i.e. we will create an end item of type General Purpose.

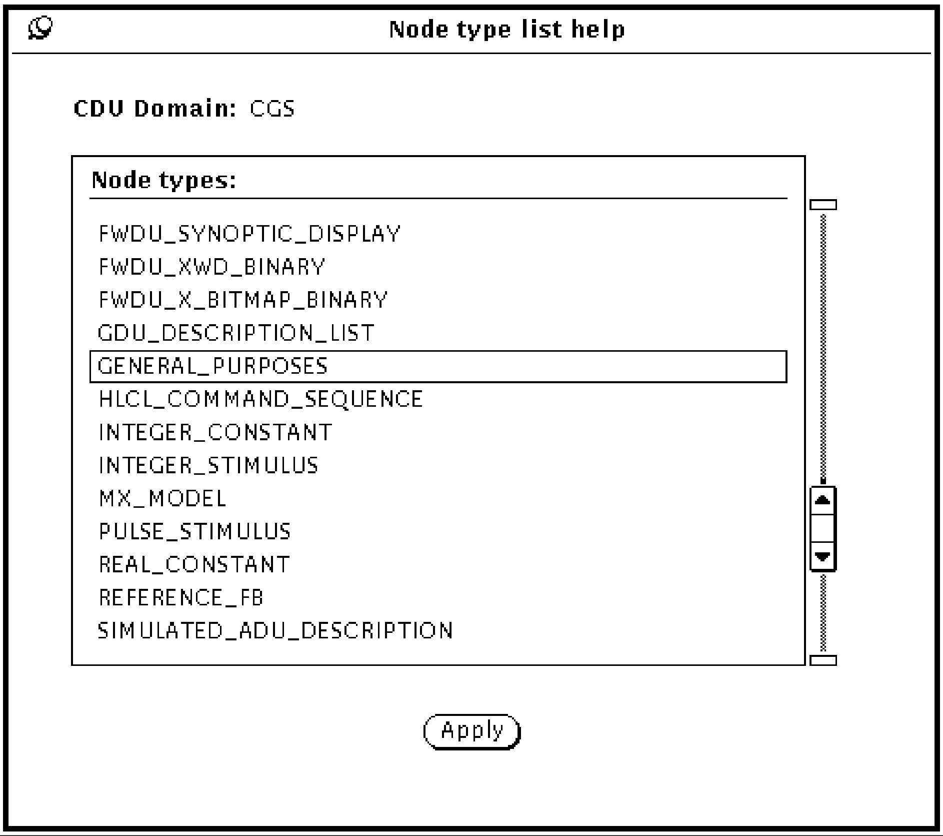


Figure 6–9 : *Node type list help*

Creating an End Item (exemplary of type General Purpose)

- **Navigate** to a user tree node as described in section 6.1.2.2.
- **Select File–>Create Node...** in the command part of the I_MDB Navigator window. This opens the ***Create user tree node*** box (see Figure 6–7).
- **Enter** the *<name>* behind the entry **Name.**
- **Click** on ***Type...***. button. The ***Node type list help*** box listing all end item types associated to the CDU domain (see Figure 6–9) appears. For this procedure the type GENERAL_PURPOSES is selected.

continued ...

- **Click** on the a.m. end item type and click on the ***Apply*** button in the ***Node type list help*** box. The node type 'virtual' in the ***Create user tree node*** box is now replaced by GENERAL_PURPOSES.
- **Enter** a Configuration Item number, if applicable. (Configuration Items (CI) are used to identify system components formally to be delivered.
- **Click** on the ***Owner...*** button. This will provide the ***User List*** box, listing all known user of the MDB instance.
- In the ***User List*** box, **click** on the user to be selected as owner followed by a **click** on the ***Apply*** button. Now the user name is displayed behind the ***Owner...*** button in the ***Create user tree node*** box.
- **Enter** some <*description text*> behind the field ***Description :***.
- **Click** on the ***Apply*** button of the ***Create user tree node*** box. Now the latter box disappears and the AP is listed as part of the Node-List of the I_MDB Navigator window

SInce now the new end item is identified, the next step is to define the contents of the end item. The creation of data and thus the available operations are strongly dependent on the end item type. However, there are still some general operations available, as explained in Table 6–1, which apply for all end items and even for virtual nodes.

Operation	Description
Open	In case several data are to be specified directly on the end item, the Open operation provides further sub-menues. In case of a virtual node, Open navigates to the next lower level.
Pathname & SID...	Show the end item Short Identifier (SID) and the corresponding pathname, but can also be used to view the pathname of a specified SID.
Tools	Depending on the End Item Type an appropriate tool (program) is invoked. Using this tool the data of an end item are created / modified. (see also the following sections)
Delete	Removes the node /end item from the current CDU version.
Copy	Copies the node /end item. The copy functions works also on sub-trees in case that a virtual node is further refined.
Paste	Pastes a copied node / end item into the current CDU version.

Table 6–1 : List of General Operations on Nodes including End Items

The access of the operations is quite easy and shown in the following procedure. In this case it is a general description not based on the above example, except, that the exemplary shown display of end item operations (see Figure 6–10) is based on it.

Not described inhere, are the several tools and data entry boxes, because, again, these are depending on the end item types and are either described in other parts of this manual or can be derived from the data description in appendix 7.6.2.

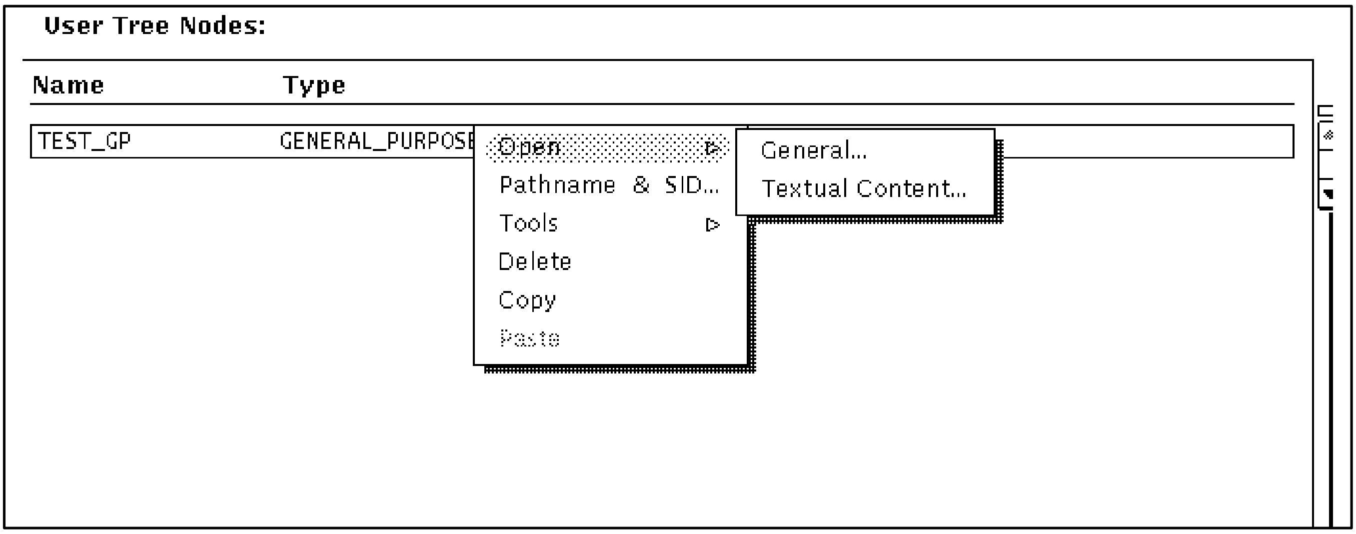


Figure 6–10 : *Exemplary shown End Item operations with the I_MDB Navigator window*

Accessing Operations on End Items.

- **Navigate** to the node you want to access.
- **Click** on the to be accessed node of the Node-List in the ***I_MDB Navigator*** window.
- **Hold** the **right mouse button**. A ***menu box*** appears on the screen.
- **Move** the **mouse cursor** to the small **arrow** behind the ***Open*** operation. This will provide you with a ***sub-menu box***.
- **Select** an operation of the ***sub-menu box***. Here the operations are different for each End Item type.
- **Release** the mouse button. This will provide you with the corresponding data entry box.
- In case another window is opened which does not allow for direct data entry, **use** the ***menu buttons*** in an appropriate way.
- **Enter** the data to the corresponding fields as described in appendix 7.6.2.

Note, if End Item data entry masks provide data of type STRING, be careful with copy / cut and paste of text, because if text is copied longer than the allowed size, the CGS tool (MDA) will exit with an oracle error message.

6.2.3 Defining CLS related items

6.2.3.1 Introduction

For definition of items of several end item types a tool named ”CLS Editor” is used. It processes items of type UCL_AUTOMATED_PROCEDURE, UCL_USER_LIBRARY, UCL_SYSTEM_LIBRARY, HLCL_COMMAND_SEQUENCE, CPL_SCRIPT, EGSE_xxx_DERIVED_VALUES and items which have parameter lists but no execution source code.

Automated Procedures (APs) are software programs written in the User Control Language (UCL) as defined in the UCL Reference Manual [2.3.3]. UCL acts as a pure programming language: automated procedures and libraries are edited off–line, compiled and kept in the database for later on–line execution.
With such programs command sequences used on the flight system or test procedures used in the check-out phase can be automated.

Thus it provides also a library concept, whereby there is a fixed **UCL System Library**, providing general functions and procedures in UCL and the possibility of **UCL User Libraries**, which can be created and used by end-users of CGS. In the UCL User Libraries the end user can define their own functions and procedures as used by various automated procedures.

The formal parameter List definition for an item must also be written in UCL as defined in the UCL Reference Manual [2.3.3].

The expressions for derived values must also be written in UCL as defined in the UCL Reference Manual [2.3.3] with restrictions defined in chapter 7.6 (description of enditem types EGSE–xxx–DERIVED_VALUE)

In contrast HLCL command sequences must be written in the High Level Command Language (HLCL) as defined in the HLCL Reference Manual [2.3.4].

CPL scripts must be written in the Crew Procedure Language (CPL) as defined in the CPL Reference Manual [2.3.5]. The Editor’s CPL Compiler generates CPL scripts in on–board format (also defined in the CPL Reference Manual [2.3.5].

Note that all mentioned languages are very UCL like.

The following sections describe the overall purpose of APs and a description is given how to modify APs. The general editor’s facilities are described in chapter 6.2.3.3. Processing automated procedures can be regarded as a typical example for using the CLS editor. Items of other end item types are modified in a similar way so that only the differences to developing APs with the CLS Editor are described in further subsections.

The CLS editor is in fact not only an editor, but an integrated development environment, which also allows for other tasks like compilation, debugging and reporting.

6.2.3.2 Purpose of APs

An automated procedure (AP) constitutes a low level automated control of the subsystem and payload of space system configurations. The objective of an AP is to execute particular subsystem or payload functions. Thus an AP is a computer program or program component, performing a sequence of operations that would otherwise be executed by a human operator. It contains verifiable, reusable processing statements necessary to perform specific operations. Normally APs will remain fixed and will be executed in the frame of action processing. In contingency cases an AP may be modified by the flight crew or ground personal.

The afore described AP is usually called Flight Automated Procedure (FLAP). The second kind of AP is known as Ground Test Automated Procedure (GTAP) and is used during the integration and check-out

phases of sub-systems in order to allow test executions automation. In this case the command sequences of such an AP have to regarded as test procedure steps (see section 8.3.1.2 for more information).

In general APs are written in **User Control Language (UCL).** Although the UCL basic syntax is that of a general–purpose programming language (derived from Modula–2), UCL is a dedicated test and operations language for monitoring and control of spacecraft sub-systems. UCL is a procedural language representing the set of all commands or instructions that can be predefined and stored, beside in APs, also in user libraries (end item type: UCL_USER_LIBRARY) in the Mission Database (MDB).

A third End Item type called UCL_SYSTEM_LIBRARY represents a system library, which provides the basic set of functions and procedures for AP development. These libraries are fixed and shall not be modified by end users, but can be used for the purpose of re-compilation.

6.2.3.3 How to Develop APs

The first task to undertake is the creation of an end item of type UCL_AUTOMATED_PROCEDURE or UCL_USER_LIBRARY. This can only be done within CDU versions, which are of one of the domains CGS, EGSE or UCL_LIBRARY as described in Appendix–7.6.2. Thus the procedure for the AP development is as follows:

Creating an automated procedure

- **Navigate** to a user tree node as described in section 6.1.2.2.
- *Remember that the CDU version you selected is of domain CGS, EGSE or UCL_LIBRARY.*
- **Select *File*→ *Create Node...*** in the command part of the main window. This opens the ***Create user tree node*** box (see Figure 6–7).
- **Enter** the *<AP–name>* behind the entry ***Name*.**
- **Click** on ***Type...***. This opens the ***Node type list help*** box listing all end item types associated to the CDU domain. For example in Figure 6–9 the domain is CGS, where you can select the type UCL_AUTOMATED_PROCEDURE.
- **Click** on the a.m. end item type and click on the ***Apply*** button in the ***Node type list help*** box. The node type 'virtual' in the ***Create user tree node*** box is now replaced by UCL_AUTOMATED_PROCEDURE.
- **Enter** a Configuration Item number, if applicable. (Configuration Items (CI) are used to identify system components formally to be delivered.
- **Click** on the ***Owner...*** button. This will provide the ***User List*** box, listing all known user of the MDB instance.
- In the ***User List*** box, **click** on the user to be selected as owner followed by a **click** on the ***Apply*** button. Now the user name is displayed behind the ***Owner...*** button in the ***Create user tree node*** box.
- **Enter** some *<description test>* in the field ***Description***.
- **Click** on the ***Apply*** button of the ***Create user tree node*** box. Now the latter box disappears and the AP is listed as part of the Node-List in the I_MDB Navigator window.

The above procedure works the same way for all database items that can be processed with the CLS editor. Only the operation set for items of other end item type may differ from the operation set for automated procedures. Figure 6–11 shows that, apart from the common operations, the only available operation on an automated procedure is the invocation of the CLS editor and compiler.

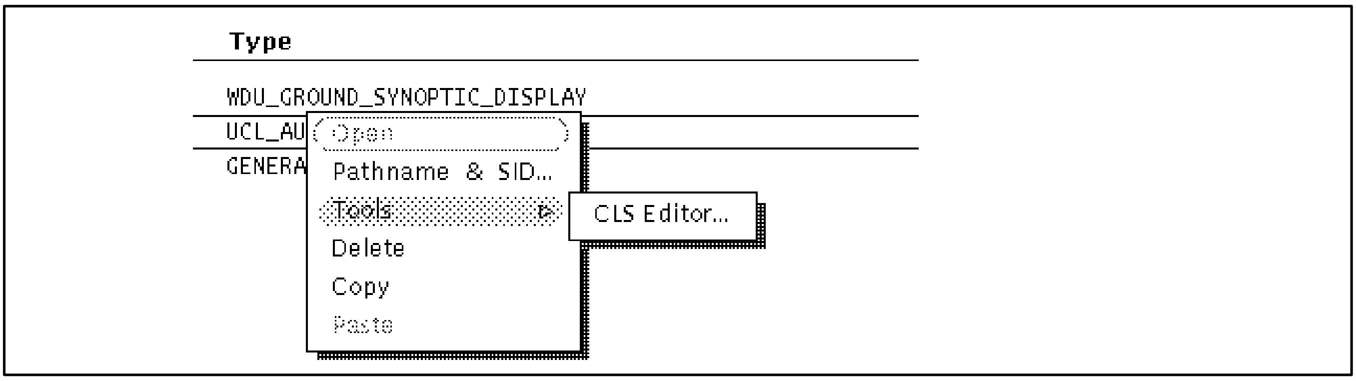


Figure 6–11 : *Operations on end items of type Automated Procedure*

The following procedure describes how to access an automated procedure. Database Items of other types are accessed in a similar way. Major differences for them are described below.

Accessing automated procedures

- **Start** the ***I_MDB Navigator*** as described in section 6.1.2.1 in procedure ”Starting a Mission Configuration Session”.
- **Navigate** to the automated procedure you want to edit (see 6.1.2.2).
- **Select** the automated procedure and **press** the right mouse button (a menu appears).
- **Select** ***Tools*** → ***CLS editor*** and **release** the mouse button. After a few seconds the CLS editor appears on the screen (see Figure 6–12).

The I_MDB tool will display a message that the CLS editor has been started in *batch mode*. This means that you can work with I_MDB and one ore several CLS editors in parallel. I_MDB will **not** go into a busy state and wait until you have finished your work with the CLS editor.

After the above mentioned actions the CLS editor will come up with its main window. How this looks like is shown in Figure 6–12.

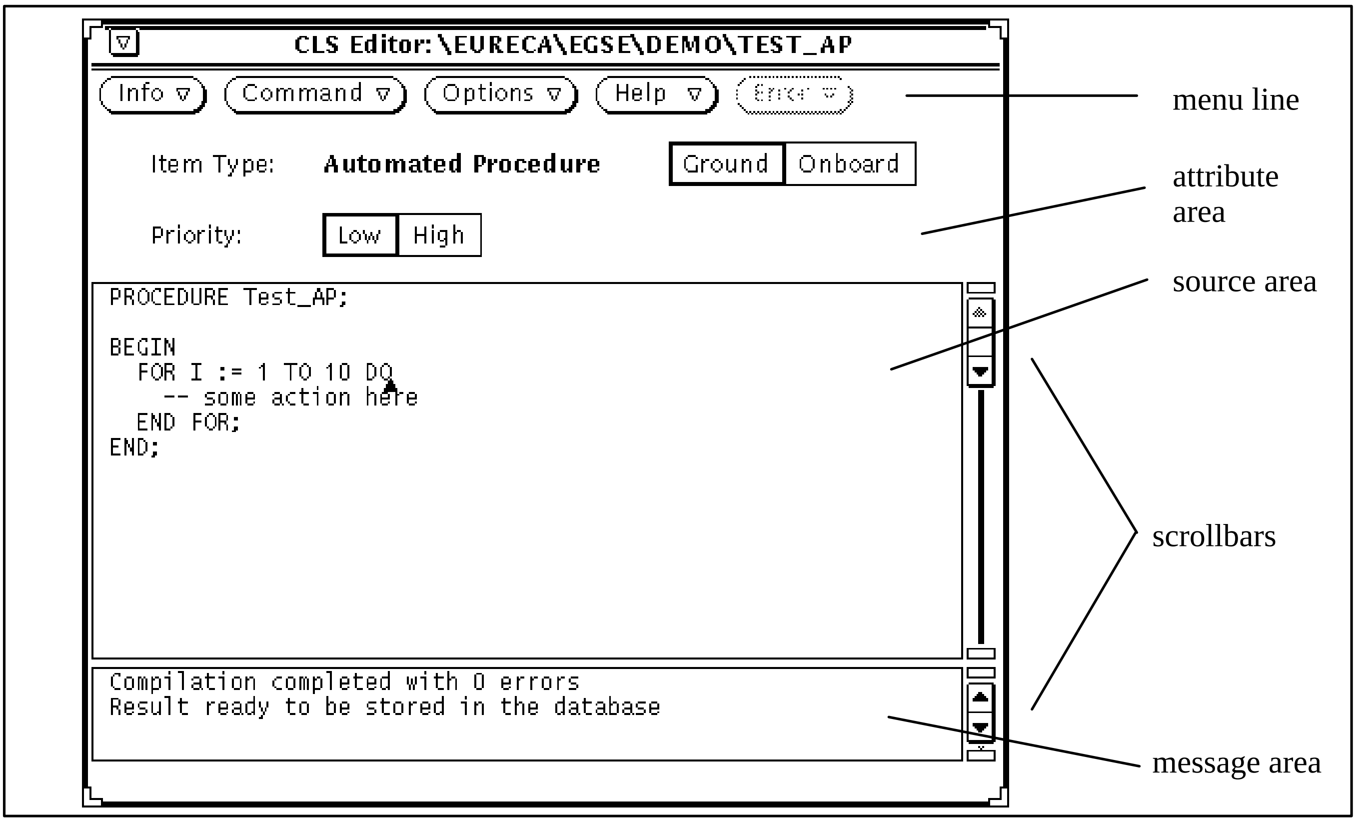


Figure 6–12 : *CLS Editor with UCL Compiler*

Now you can work with the CLS editor and generate or modify the required AP. The general editing facilities in the source area work very similar to the normal SUN OS ”textedit” editor.

Compiling an AP

- **Click** with the **left mouse button** on the ***Command*** button.
- Or **press** the ***Command*** button with the **right mouse button** and select ***Command*** → ***Compile***.

Compilation messages are displayed in the message area.

If there are compilation errors the error button is enabled. Using the error menu you can localize the erroneous source text positions. The text caret is positioned at an error location and the appropriate message is displayed on top of the message area (see Figure 6–13).

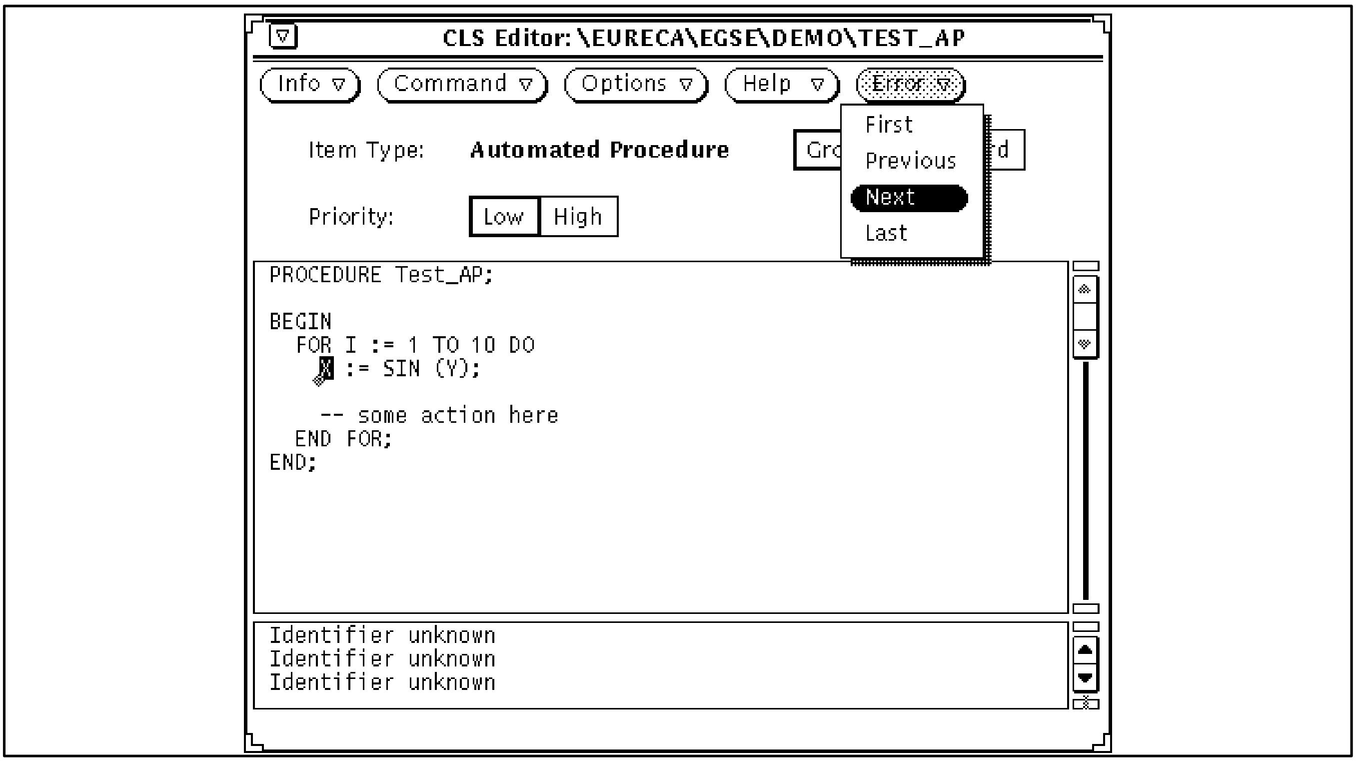


Figure 6–13 : *CLS Editor: Error menu*

Finding error locations

- **Click** with the **left mouse button** on the ***Error*** button to get to the next error location.
- Or **press** the ***Error*** button with the **right mouse button** and select ***Error→ Next***.
- To localize the first, previous or last error **press** the ***Error*** button with the **right mouse button** and select ***Error→ First***, ***Error→ Previous*** *or* ***Error→ Last***.

You can control the compiler's behaviour in some way. For this purpose use the ***Options*** menu. First of all you can generate a compiler listing.

Generation of a listing

- **Click** with the **left mouse button** on the ***Options*** button.
- Or **press** the ***Options*** button with the **right mouse button** and select ***Options→ Listing..***.

The CLS editor's listing window will come up. It is shown in Figure 6–14.

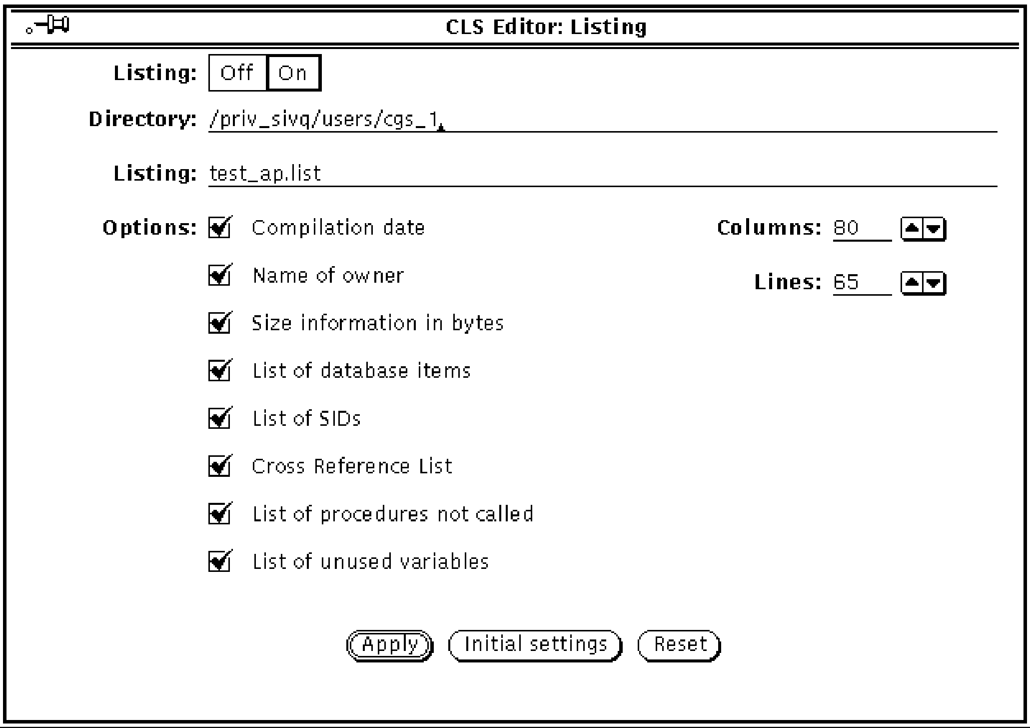


Figure 6–14 : *CLS Editor: Listing Window*

Several listing properties can be set in the listing window. To really get a listing the ***Listing*** choice on top of the listing window must be switched to *on* – if it is not already on – and the changes must be applied with clicking on the ***Apply*** button. A second possibility is to click on the ***Set default*** button to get the listing default values. The listing window in the figure is one with its initial default values. After clicking on the ***Set default*** button (which only lets the listing window display the default values) you must click on the ***Apply*** button to make the default values applicable. The third possibility is to click on the ***Reset*** button. This action restores the last “applied” values. Again, click on the ***Apply*** button to make them applicable.

Note that for each (from I_MDB invoked) CLS editor the listing window’s initial values are the default values.

As further option the CLS editor provides a make mode in which the current compilation unit (the AP) and all compilation units (e.g. UCL user libraries) the AP depends on are compiled automatically if necessary.

Make mode

- **Press** the ***Options*** button with the **right mouse button** and select ***Options***→ ***Make..***.

A simple small window is displayed where you can choose between ***on*** and ***off***. The make mode can be performed on three different display levels.

- **Select** one of Show: ***All***, ***Erroneous*** or ***None***.

The default is to display a separate editor window for a compilation unit only in case of compilation errors (Show: ***Erroneous***). It is also possible to compile and store the units in background (Show: ***None***). In this case it is necessary generate an error list file (see below) to get compilation error messages. The third possibility is to step through all units that have to be compiled (Show: ***All***). For each unit to be compiled a separate editor window comes up and the compilation has to be done manually by selecting ***Command***→ ***Make*** (or

clicking with the left mouse button on the Command button). In make mode this entry replaces the ***Command*** → ***Compile entry***. Store the item afterwards and quit the window. A new editor window will come up with the next unit or the compilation will continue with the previous unit. During compilation of units other than the starting unit the base editor window is deactivated so that even in this mode a correct compilation order is ensured.

Note that in case of cyclic dependencies between units (e.g. AP A calls B and B calls A) the make process terminates with an appropriate error message.

The CLS editor can be forced to generate an error list file containing essential information about the compilation process in a short form. The file's contents is the compilation unit name and a list of compilation error messages (in make mode for each compiled unit).

Generation of an error list file

- **Press** the ***Options*** button with the **right mouse button** and select ***Error List..***.

The compiler can be run with or without optimization.

Choosing the compiler's optimization mode

- **Press** the ***Options*** button with the **right mouse button** and select ***Options*** → ***Optimize***...

A simple small window is displayed where you can choose between ***on*** and ***off***. The default is *optimization on*.

At any time you can store the automated procedure which you are processing.

Storing an automated procedure

- **Press** the ***Command*** button with the **right mouse button** and select ***Command*** → ***Store***.

If your compilation was not successful and you try to store the automated procedure, the CLS editor will ask you for confirmation because in this case only the source can be stored. If there is already compiler generated data in the database from an earlier successfully compiled AP version the store operation will lead to loss of that data because the old generated data is inconsistent with the current AP source code. Figure 6–15 shows how the conformation request looks like.

If you don't want to store the item – in this case only the source code – just click on the dismiss button of the confirmation window and the item contents in the database will remain unchanged.

On execution of the store operation all item attributes (as given in the CLS editor's item attribute area) are always stored – independent of the item's compilation state. If you want to store the AP and quit the editor afterwards you can do it by simply selecting ***Command*** → ***Store & Quit***.

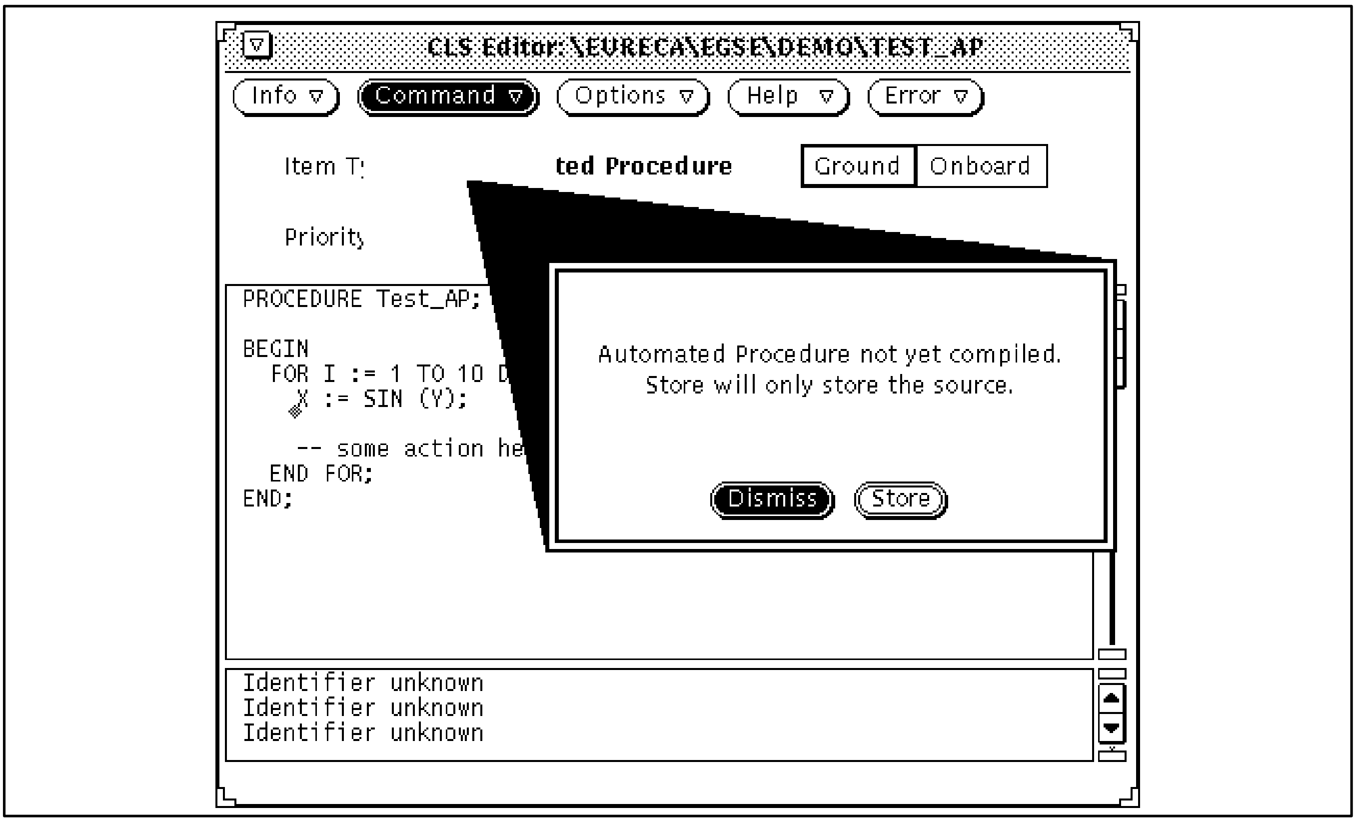


Figure 6–15 : *CLS Editor: Confirm the store operation*

While you are working with the CLS editor some history information can be obtained about the user's actions.

Viewing the action log

- **Press** the ***Comamnd*** button with the **right mouse button** and select ***Command→ View Log***.

A window comes up which lists the store and compile actions performed since editor startup. This window is especially interesting if a unit has been compiled in make mode without generation of an error list file. A list of the processed units can be obtained from here.

Furthermore while you are processing an automated procedure you can obtain some information about it.

Getting information about an AP

- **Press** the ***Info*** button with the **right mouse button** and select ***Info→ Item***.

An item information window comes up and displays some of the item's attributes, see Figure 6–16.

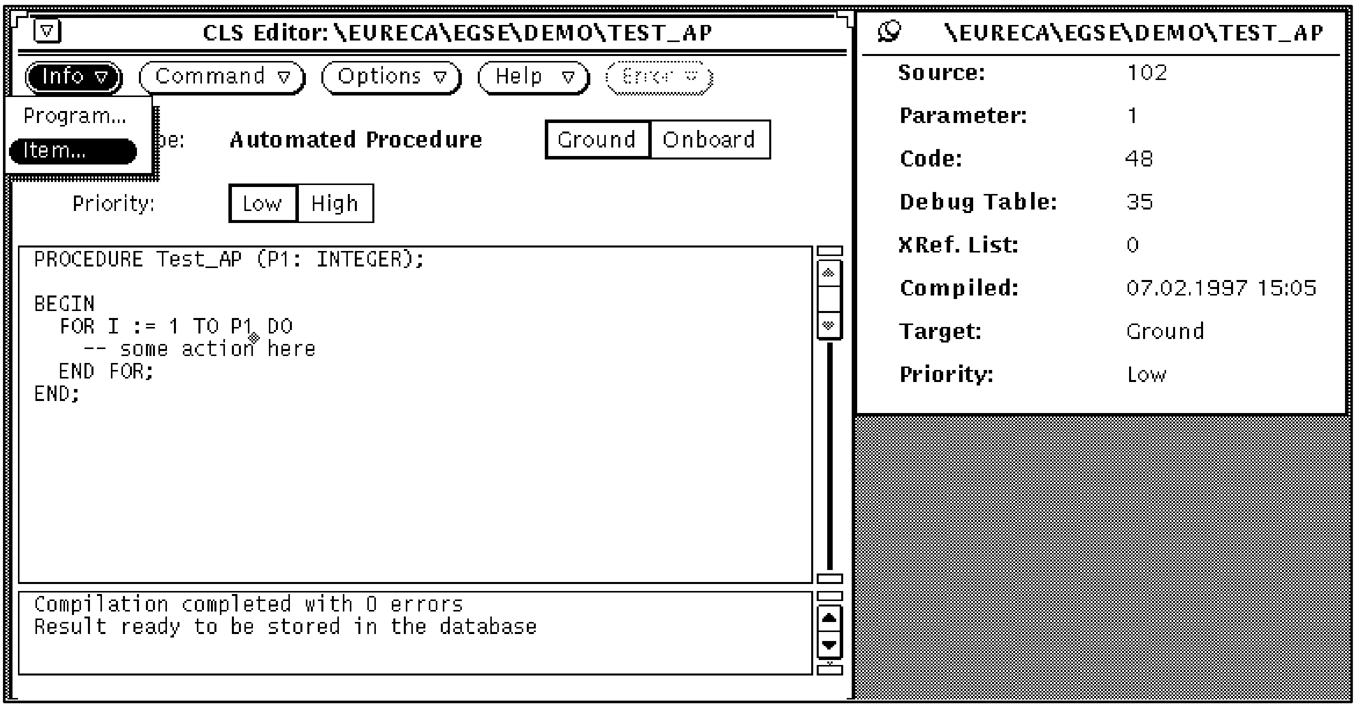


Figure 6–16 : *CLS Editor: Item information window*

The item information window for an automated procedure comprises the following attributes:

- size of the source code in characters
- number of parameters
- size of compiler generated code in bytes
- compilation date
- size of the compiler generated debug table in bytes
- number of entries in the cross reference list
- onboard flag (also shown in the CLS editor's main window)
- priority (also shown in the CLS editor's main window)

If the item information window is on the screen while you modify the automated procedure the changes are recognized and the information window will be updated within a short period of time (normally one second). You can simply observe this by typing a character in the source area. The character count in the information window will be incremented.

For an initially loaded AP (i.e. an unmodified AP) the item information window shows attribute values reflecting the item contents in the database.

The ***Info*** → ***Program*** button can be used to obtain some global information about the CLS editor such as an enumeration of the different compilers it comprises.

If you need help on UCL syntax during your work you can get some from the CLS editor in the following manner.

Getting syntax help

- **Click** with the **left mouse button** on the ***Help*** button.
- Or **press** the ***Help*** button with the **right mouse button** and select ***Help*** → ***Syntax...***

The CLS editor’s syntax help window as shown in Figure 6–17 will be displayed on your screen.

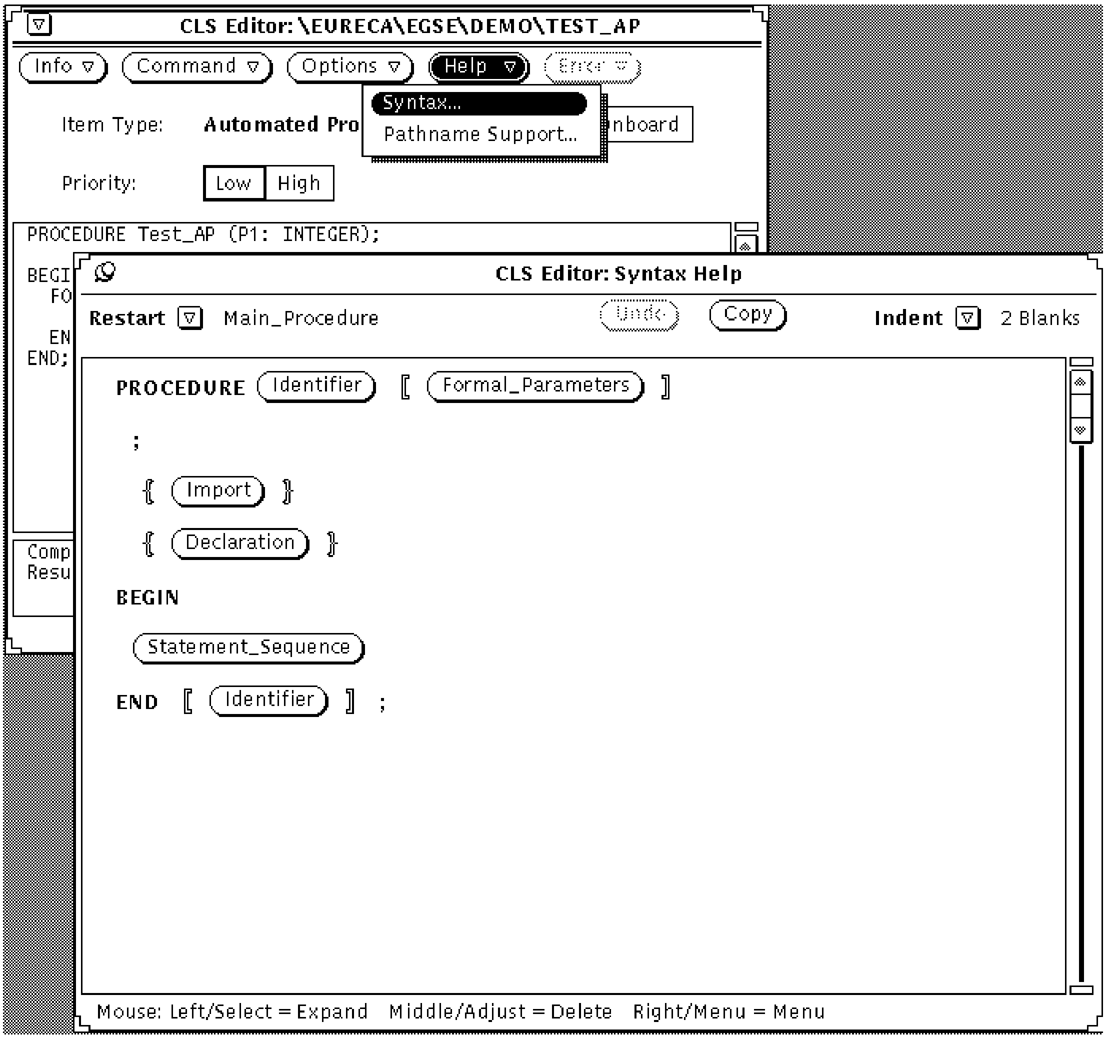


Figure 6–17 : *CLS Editor: Syntax Help Window*

Within the help window you can get a list of all UCL commands as shown in Figure 6–18 by **holding** the **right mouse button** and moving the mouse cursor to the **arrow field** behind ”**restart**”. If you select a command, you will get the syntax listed in the help window. Figure 6–18 shows the choices to restart the syntax help.

Using the ***copy*** button in the help window you get the appropriate command skeleton as part of your UCL program.

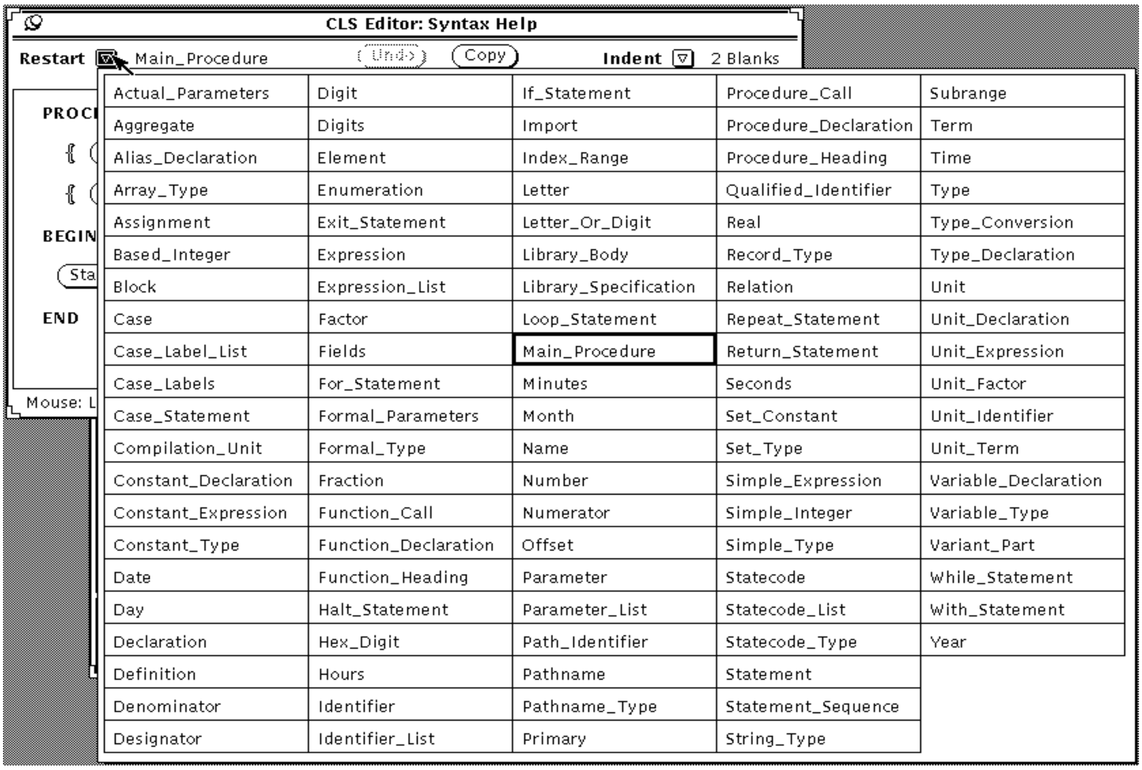


Figure 6–18 : *CLS Editor: Syntax Help – Restart choices*

Help on names of end items during work with the CLS editor can be obtained in the following manner.

Getting pathname help

- **Press** the ***Help*** button with the **right mouse button** and select ***Help*** → ***Pathname Support***...

The pathname support window comes up and allows you to look for existing end items. The selected name will be copied into the editor’s source text window at the text caret position when clicking on the ***Apply*** button. For *each* pathname to be inserted in the source the action sequence ”selecting ***Help*** → ***Pathname Support***, clicking on the ***Apply*** button” must be performed. For a detailed descripton of the pathname support window please refer to the corresponding detailed user manual [2.1.1.3].

When you finished your work want to quit the CLS editor it checks if you modified the automated procedure but didn’t store the changes yet in the database. In this case you will be asked to execute a store operation or to intentionally discard the changes as shown in Figure 6–19. You also have the possibility to cancel the quit operation.

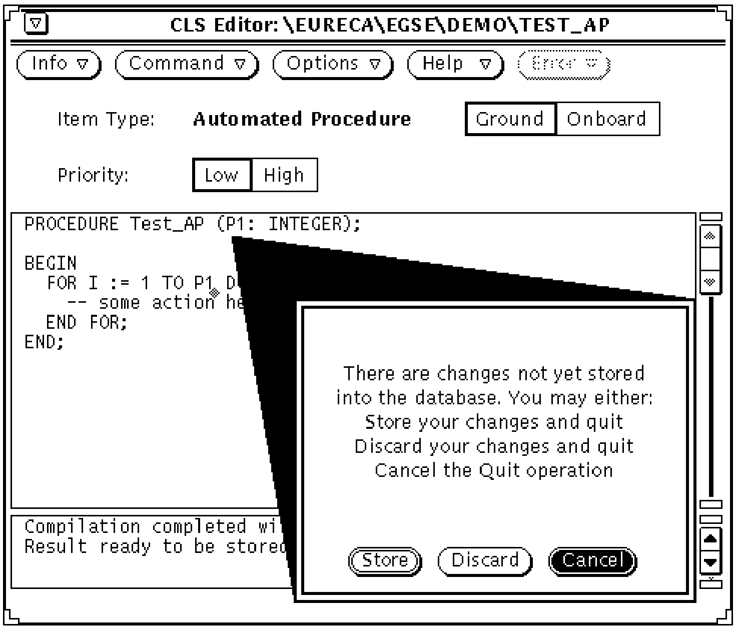


Figure 6–19 : *CLS Editor: Quit — Confirmation request*

6.2.3.4 Developing a User Library

For processing end items of type User Library as for APs a domain of type CGS, EGSE or UCL_LIBRARY must be chosen. The major difference with respect to the usage of the CLS editor is that in the source area as one choses the user library specification or the user library body is displayed. You can toggle between them by clicking on the spec/body choice of the CLS editor's main window, see Figure 6–20. On the left side the editor's window is shown with the specification loaded and on the right side the editor has loaded the user library body.

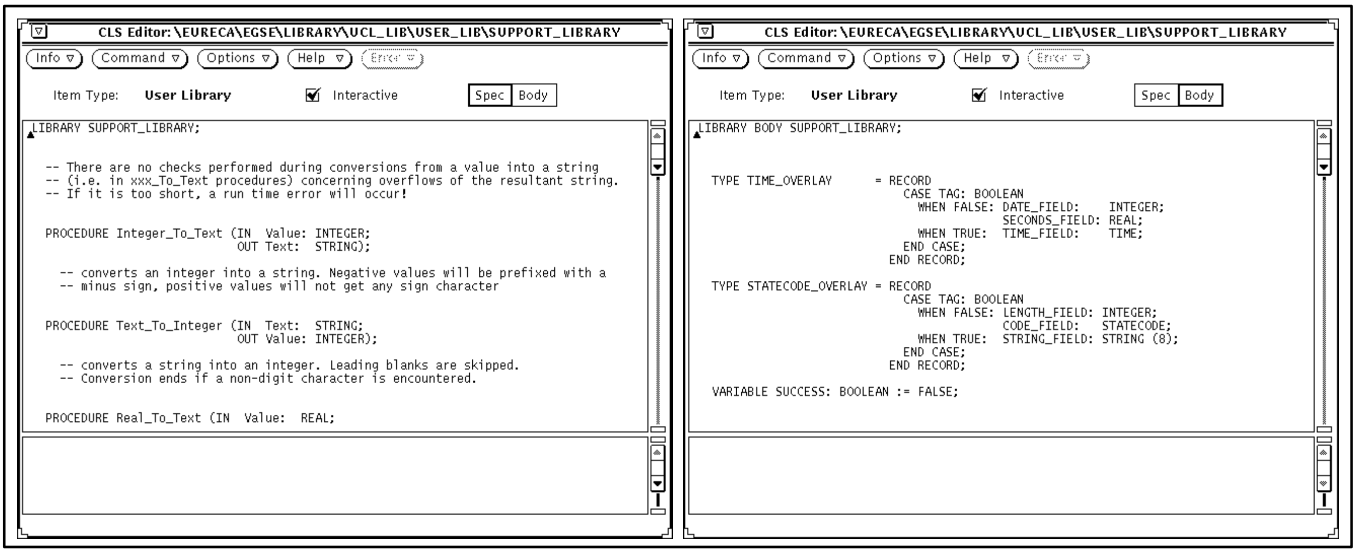


Figure 6–20 : *CLS Editor: Processing a user library*

The specification and the body source are processed in a similar way to automated procedures. But when you try to compile the user library source the editor will force you to consider the correct compilation order. If

you try to compile a user library body and the specification is not already compiled you will get a hint to please compile the specification first, see Figure 6–21.

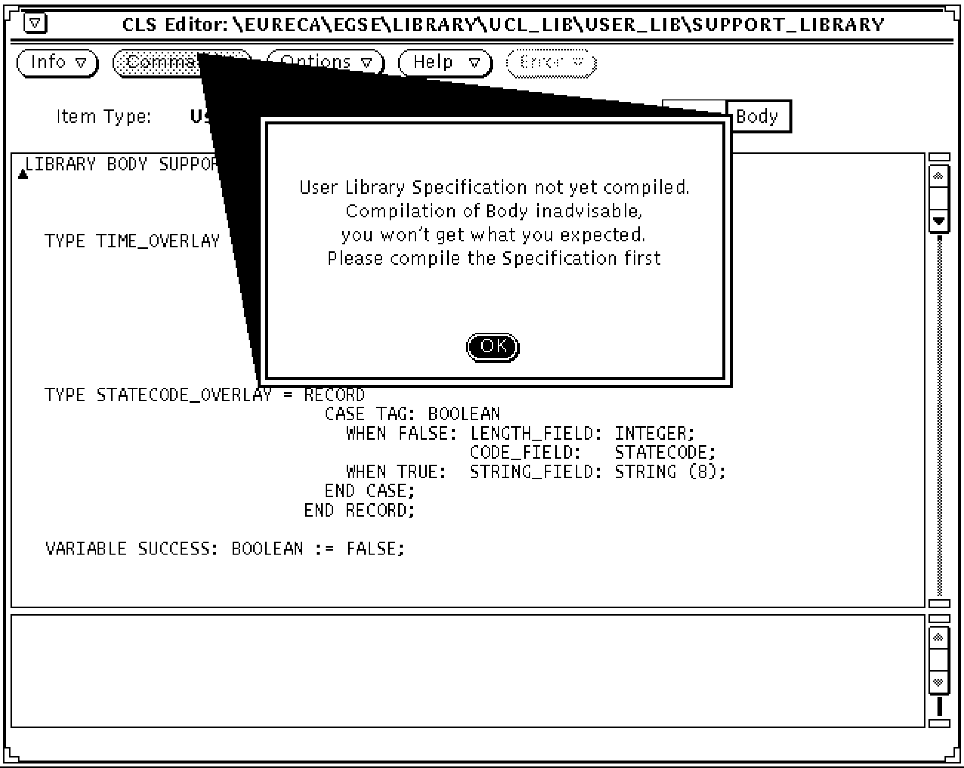


Figure 6–21 : *CLS Editor: forced compilation order for a user library*

The item information window for a user library comprises the following attributes:

- size of the specification and body source code in characters
- size of compiler generated code in bytes
- size of compiler generated symbol table in bytes, generated during specification compilation
- size of the compiler generated debug table in bytes, generated during body compilation
- number of entries in the specification and body cross reference list
- compilation date
- interactive flag (also shown in the CLS editor's main window)

6.2.3.5 Developing a System Library

The CLS editor's usage for system libraries is nearly the same as for automated procedures.

Here again only *one* source is loaded into the source area, i.e. the specification of the system library.

The item information window for a system library comprises the following attributes:

- size of the source code in characters
- size of compiler generated symbol table in bytes
- number of entries in the cross reference list
- compilation date

- interactive flag (also shown in the CLS editor’s main window)
- system library ID (also shown in the CLS editor’s main window)

The system library’s ID depends on the implementation of the system library body. It must be set correctly to calls to system library procedures work correctly. The default value 0 is an invalid ID and if you try to store the system library specification with this ID you will be asked for confirmation.

6.2.3.6 Developing an HLCL Sequence

An HLCL Command Sequence may comprise simple HLCL commands and HLCL compound statements such as if-then-else or loops. A detailed description is given in the HLCL reference manual in document 2.3.4. HLCL is syntactically nearly the same as UCL. For this reason the HLCL reference manual is an extension to the UCL reference manual.

HLCL sequences can be stored in files (see also *Invocation Interface*, chapter 6.2.3.10 below) or in the database. The latter ones are processed with the CLS editor in a very similar way to automated procedures, but for HLCL sequences only a domain of type CGS or EGSE can be chosen. The end item type name is HLCL_COMMAND_SEQUENCE.

The main difference is that for HLCL sequences the operation *compile* has a different meaning. HLCL sequences are executed with the HLCL interpreter which accesses the source code. So no compiler generated code is necessary. But nevertheless you may *compile* an HLCL sequence in the sense that the sequence is **checked** for syntactical errors.

An HLCL check is significantly fewer than a UCL compilation. E.g. no checks are made (and cannot be made) if used pathnames are really names of existing end items. Also no checks can be made if used variables are declared. Variables are rather variables of an interactive HLCL session than local objects of a sequence. So it is sufficient to declare them immediately in an HLCL session before executing the sequence.
If objects used in an HLCL sequence are not defined when you *execute* the sequence you will receive error messages from the HLCL interpreter.

The item information window for an HLCL sequence comprises the following attributes:

- size of the source code in characters
- number of parameters
- size of “compiler generated code” in bytes, i.e. information whether the sequence has been checked or not. (0 for not checked, 1 for checked)
- check date

Creation of an HLCL Sequence in the Unix file system:

To create or edit an HLCL Command Sequence in the Unix file system, follow these steps:

Create HLCL Command Sequence in Unix file system

■ Call the CLS Editor from a Unix shell:
$CLS_HOME/bin/sun5/cls_editor <*filename*>.hlcl <*options*>

 If the sequence file does not exist the CLS Editor asks for a creation confirmation, see Figure 6–22.

 Answer OK if you want to create the file or Cancel, if you want to quit the CLS Editor immediately.

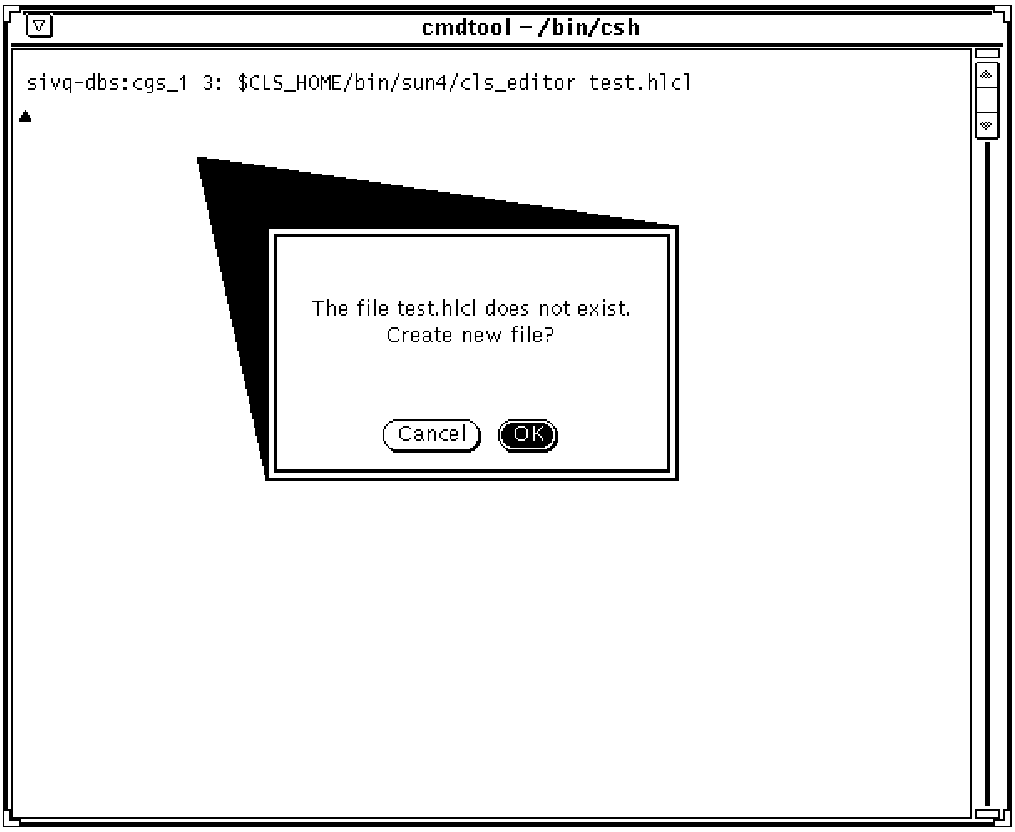


Figure 6–22 : *CLS Editor asking for confirmation of file creation*

Edit HLCL Command Sequence in Unix file system

- After confirmation or if the sequence already exists, the CLS Editor window pops up.
- Now you may enter or edit the HLCL Command Sequence the same way as any editing is performed in the CLS Editor.
- If you want to store your changes in the file, leave the CLS Editor via the **Command–>Store & Quit** menu option (see Figure 6–23).
- Otherwise, use the **Quit** option of the window menu.

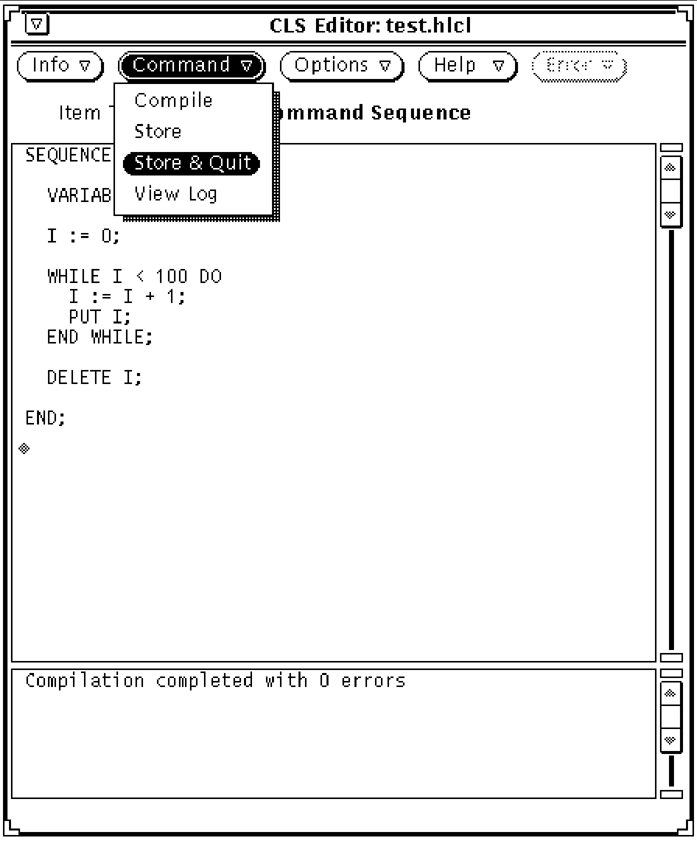


Figure 6–23 : *HLCL Command Sequences in the Unix file system can be edited with CLS as well*

6.2.3.7 Developing a CPL Script

The APM onboard LAPTOP provides the capability to execute crew procedures. These procedures can be monitored and controlled by the crew member during execution. They are presented in a readable way so that any step of the procedure is understood by the crew member.

The language which constitutes the presentation of crew procedures to the crew as well as the language of the source code to be developed is the CPL (Crew Procedure Language). The major means how a CPL script influences the behaviour of the APM is the commanding of the software which controls the system.

These commands to the onboard software are called software commands. They are defined in the Mission Data Base and can be used in CPL scripts and other sources like Onboard Synoptic Displays.

The basic entity is the script. It contains elements as defined by the script language (for more information about CPL refer to 2.3.5).

The script language has four different representations:

- the ”source code” representation is the ASCII text as entered by the developer using the Crew Procedure Language,
- the ”processing code” representation is the result of the compiling process. This is the code given to the onboard interpreter for processing,
- the ”presentation code” is a subset of the script. This is what the crew will see during crew procedure operations,

- the logic flow diagram (LFD) shows the script as a flow diagram where specific language constructs are replaced by graphical items. Remaining text is displayed as ”presentation” code.

CPL scripts are processed with the CLS editor in a very similar way to automated procedures. For processing end items of type CPL script a domain of type CGS or SDDF must be chosen.

The item information window for a CPL script comprises the following attributes:

- size of the source code in characters
- size of compiler generated code in bytes
- number of entries in the cross reference list
- compilation date

6.2.3.8 Developing an item with parameter list

The CLS editor’s usage for items which have parameter lists is nearly the same as for automated procedures.

The item information window for these items comprises the following attributes:

- size of the source code in characters
- number of parameters
- size of compiler generated symbol table in bytes
- number of entries in the cross reference list
- compilation date

The parameter lists of some items must obey some constraints. Please refer to Chapter 7.6.2 and document [2.3.2] for a description of these constraints.

If the CLS editor is invoked for a newly created item it will generate source for an initial parameter list and display it in the source window. In general this created source represents an empty parameter list, i.e. just a pair of parenthesis. But if the parameter list constraints are so rigid that they lead to a fixed parameter list form, source for a non empty parameter list obeying the given constraints is generated. E.g. the latter case applies if the parameter list constraints only allow one parameter, which must be of mode in, of type real and without default value.

6.2.3.9 Developing an Expression for Derived Values

The CLS editor’s usage for derived value items is nearly the same as for items with parameter lists.

Expressions are used to describe the calculation of values from other values (enditems of type EGSE_XXX_MEASURMENT, EGSE_xxx_SW_VARIABLE or EGSE_xxx_DERIVED_VALUE).

Theexpressions must follow UCL syntax and obey some constraints. Please refer to Chapter 7.6.2 for a description of these constraints.

The item information window for these items comprises the following attributes:

- size of the source code in characters
- size of compiler generated symbol table in bytes
- number of entries in the cross reference list (i.e. the number of enditems used in the expression)
- number of entries in the SID list (i.e. number of SIDs used in the generated code)
- compilation date

6.2.3.10 CLS Editor Invocation Interface

The CLS Editor is a UNIX process named `cls_editor`. It is to be started with command line parameters and options. The process returns a completion code. In interactive mode the completion code is always 0. In batch mode it indicates compilation success/failure:

0 = no compilation errors,
1 = compilation errors.

Depending on its type, a compilation unit may be contained in the database or in a file (the file may be a named pipe). When contained in the database, the attributes of the database item determine the type of compilation unit. When contained in a file, the type of compilation unit is, by default, determined by the file name suffix:

`.hlcl`	HLCL command sequence
`.fwdu`	FWDU command table

If the file name suffix does not uniquely define the type, it must be explicitly given with the `–unit` option.

Option names and keywords are case–insensitive. They may be uniquely abbreviated. Compound names may also be abbreviated by abbreviating the single components (e.g. `–error_list` may be abbreviated as `–err_li` or `–err`).

Call:

cls_editor	<pathname>	<options>	(call for database item)
cls_editor	<filename>	<options>	(call for file)

Options:

–environment <environment>

defines the database user environment. If omitted, the environment will be obtained from the environment variable `MDA_ENVIRONMENT` in the same syntactic form:

<environment> ::=
CCU <element_config> <mission> <system_tree_version> <ccu> |
CDU <element_config> <mission> <system_tree_version> <cdu>

<ccu> ::=
<pathname> <ccu_name> <version>.<issue>.<revision>

<cdu> ::=
<pathname> <version>.<issue>.<revision> <test_version>
<instance>

Example:

–env CCU APM DUMMY_MISSION 1 \APM\TEST\GROUND\CLS CLS_TEST 1.0.0

–unit <unit>

defines the type of compilation unit contained in a file. If omitted, the type will be determined by the file name suffix (see above).

<unit> ::= HLCL_SEQUENCE | FWDU_COMMAND_TABLE

–file
–database

For the very rare case that the name given as a parameter cannot be uniquely identified as a file name or database pathname (e.g. file name starting with '\'), one of these options can be used to resolve the ambiguity.

–make

requires that the specified compilation unit as well as all units referenced by this unit directly or indirectly are to be updated. Items will be compiled, if necessary.

–fast

requires that only the specified compilation unit is to be updated. Referenced items are checked for up–to–date status, bu not recompiled.

–batch

indicates that the CLS Editor is to be called in batch mode.

–error_list [<filename>]

requires an error list to be generated in a file. For each compiled unit it contains a section of the form

```
<file_or_path_name>
        <line> <column> <error_message>
        ...
        <line> <column> <error_message>
```

The first line contains the path name or file name, resp., of the compiled unit. It starts in the first column of the the line. Each of the following lines contains a single message prefixed with the line and column number as a reference to a position in the source text. These lines are indented with two spaces. The lines are sorted by line and column numbers. If line and column are both 0, the message does not refer to the source code but reports some general event, e. g. an internal compiler problem.

For user libraries separate sections are generated for specification and body, the text "spec" or "body" will be appended to the corresponding path name header. For a compilation with the `–make` option one section per compiled unit will be generated. Sections are separated by an empty line.

If the file name is omitted, it will be derived from the input file name or the database item node name, resp.:

<name>.err

–listing [<filename> [<filename>]]

requires a listing to be generated in a file. For user libraries separate listings are generated for specification and body. If file names are omitted, they will be derived from the input file name or the database item node name, resp.:

<name>_.list for the specification
<name>.list for the body (and all non–library units)

	parameter	–environ ment	–make –fast	–unit	–file	–data base	–batch	number of files –error_ list	number of files –listing
AP	<path>	x	x			(x)	x	0 – 1	0 – 1
User Library	<path>	x	x			(x)	x	0 – 1	0 – 2
System Library	<path>	x	x			(x)	x	0 – 1	0 – 1
HLCL Sequence	<path>	x		x		x	x	0 – 1	0 – 1
	<file>	x		x	x		x	0 – 1	0 – 1
CPL Script	<path>	x	x			(x)	x	0 – 1	0 – 1
FWDU Cmd. Table	<file>	x		x	(x)		x	0 – 1	0 – 1
Parameter List	<path>	x	x			(x)	x	0 – 1	0 – 1
Derived Value Expression	<path>	x	x			(x)	x	0 – 1	0 – 1

x = option allowed
(x) = option allowed but meaningless (redundant)

Figure 6–24 : *Parameter/option combinations for various compilation units*

6.2.4 Defining Software Replaceable and Software Exchangeable Units

6.2.4.1 Purpose of SWRUs and SWEUs

SW Replaceable Units (SWRU) and SW Exchangeable Units (SWEU) are programs normally written in Ada or C, but also in other general purpose programming languages. The differences between SWRUs and SWEUs are:

- SWRU are programs running on-board. Normally SWRU does not have much (if any) interaction with the user, but are used for e.g. controlling various systems as part of the on-board Data Management System. SWRUs are part of the flight image generation process. An on-board program is meant a SWRU, if it can be replaced during a mission.
- SWEU are programs running on ground. Since SWEUs do <u>not</u> form part of the flight image generation, the executable code will not be stored into the MDB. Like for SWRUs also SWEUs shall have the property to be exchangeable.

For both categories, from now on called SW Entity, it is possible that they refer to other End Items in the MDB. For this purpose a special Ada pre-compiler is provided, which translates the pathnames to the end items into their short identifiers or into constant values. During this translation, a cross-reference-list is generated, which will later be stored into the MDB associated to a SW Entity. For the development of SW Entities a separate compilation and generation environment is available, which is external to CGS. This section provides also for the definition of the SW Entity data sets being down-loaded into the MDB.

For SW Entities special End Item types have been defined within the MDB called *SWRU* and *SWEU*. End Items of these types can be created within CDUs being of the standard domain :

- SDDF or CGS, for both types, or
- EGSE for SWEUs only.

The assignment of data to a SW Entity will mainly be performed by loading the data from the generation environment into the Mission Database. For this purpose, two approaches are possible:

- Loading in batch via UNIX script
- Load via I_MDB/ Enditem Selection

6.2.4.2 SW Entity Data Set Load Concept

At the end of the SWRU / SWEU build process performed in CM System, three files describing a SWRU / SWEU data set are created under the following file system location (which is visible from the MDB):

$GSAF_HOME/sddf/data/sw/*SWDirName*

where *SWDirName* shall be *CMVersionIdentifier*/*TargetArchitecture* (as used in the configuration control system of the generation environment).

The files within that directory are named as:

ExecutableName	for the binary code of SWRU / SWEU executable.
*ExecutableName***.att**	Attribute file for the attributes required for the MDB upload (see below). This file is mandatory.
*ExecutableName***.xref**	containing the merged X–Reference List. This file will not be generated in case where precompilation has not been performed (ie. executable does not contain precompiled end items).

The loading of a SWRU / SWEU data set from the filesystem (see above) to the Mission Database can be initiated from the following sources:

a. Through CGS/I_MDBs' flexible tool invocation interface at <u>End Item Level</u> it is possible to exactly load one SWRU / SWEU data set. To enable this, the following information are requested from the user as user input: additional information relative to the SWOP_HOME directory, (ie. $GSAF_HOME/sddf/data/sw) which are *SWDirName* and *ExecutableName* in order to determine the full UNIX pathnames to files belonging to the SWRU / SWEU data set.

b. Through CGS/I_MDBs' flexible tool invocation interface at <u>CCU level</u> or from the filesystem though a script/command line interface it is possible to a load a number of SWRU / SWEU data sets in one go (mass load). To enable this, all data sets which are subject for the loading process have to be uniquely identified together with their pathnames (ie. MDB destination) in a Load Configuration File (see later). The load configuration file name has to be identified by the user, otherwise a default load configuration file will be used instead.

6.2.4.3 How to load SW Entity Data Sets into the MDB

Prior to the load of a SW Entity, it has to be created as an End Item beneath a user tree (CDU) version (node). The user tree version has to be of a domain which comprises the SW Entity End Item type, i.e of the domain SDDF or EGSE / CGS (see above).

The creation of a SW Entity End Item has then to follow the following procedure :

Creating a SWRU or SWEU

- **Navigate** to a user tree node as described in section 6.1.2.2.
- *Remember that the CDU version you selected is of a domain as described above.*
- **Select File–>Create Node...** in the command part of the main window. This opens the ***Create user tree node*** box (see Figure 6–7).
- **Enter** the <*SW Entity–name*> behind the entry **Name.**
- **Click** on ***Type...***. The opens the ***Node type list help*** box listing all end item types associated to the CDU domain.
- Search the wished SW Entity end item type (SWRU or SWEU) using the **Scroll-Bar** and **double-click** on the **selected type** in the ***Node type list help*** box. The node type ’virtual’ in the ***Create user tree node*** box is now replaced by **SWRU** or **SWEU**.
- **Click** on the ***Owner...*** button. This will provide the ***User List*** box, listing all known user of the MDB instance.
- In the ***User List*** box, **double-click** on the user to be selected as owner . Now the user name is displayed behind the ***Owner...*** button in the ***Create user tree node*** box.
- **Enter** some <*description test*> behind the field ***Description :***.
- **Click** on the ***Apply*** button of the ***Create user tree node*** box. Now the latter box disappears and the SW Entity is listed as part of the Node-List in the I_MDB Navigator window.

For each SW Entitity to be loaded, an attribute file ExecutableName.att must exist under the same directory where the executable file is placed (see before), having the following contents (without line number):

Line	Attribute Name	Possible Values	
1	**GenElementConfiguration**	**<Element Name>**	**(*)**
2	**GenMissionName**	**<Mission Name>**	**(*)**
3	**GenSystemTreeVersion**	**<Systemtree Version Number>**	**(*)**
4	**GenCCUPathname**	**<Pathname of CCU>**	**(*)**
5	**GenCCUName**	**< Name of CCU >**	**(*)**
6	**GenCCUVersion**	**< CCU Version Number>**	**(*)**
7	**GenCCUIssue**	**< CCU Issue Number>**	**(*)**
8	**GenCCURevision**	**< CCU Revision Number>**	**(*)**
9			
10	**CMSystemName**	**PDB | <any other>**	
11			
12	**CMVersionIdentifier**	**<CM Version Id (string)> | UNDEFINED**	
13	**TargetArchitecture**	**<Target Identifier used in CMSystem**	
14	**ExecutableName**	**UNIX name for executable**	
15			
16	**SWEntityStatus**	**CM_CONTROLLED | DEVELOPMENT**	
17	**SWEntityType**	**SWEU | SWRU**	
18			
19	**XRefFileGenDate**	**<date of first precompilation: mmm–dd–yyyy hh:mm:ss>**	
20	**XRefFileChecksum**	**<ref file checksum>**	
21			
22	**ExecFileGenDate**	**<ref file generat. date: mmm–dd–yyyy hh:mm:ss>**	
23	**ExecFileSize**	**<file size in byte>**	
24	**ExecFileChecksum**	**<exec file checksum>**	

(*) Items specify the generation/precompilation environment of the executables. Items are not used for selecting configuraiton during load. Will be superseded by selected CCU/CDU Version selected in I_MDB during interactive loading resp. specification in configuration file during batch operation.

6.2.4.4 Load as Batch Operation

To load SW Entities into the MDB without user interaction, a script is available:

`$SWES_HOME/bin/sun5/load_sw_datasets –config <config_file>  [SILENT]`

which loads a set of SW entities into the MDB as specified in the <load_configuration> files.

If SILENT is specified, progress windows will be suppressed.

The “load_sw” program loads the SWRU / SWEU according to a configuration file `<config_file>.cfg` placed under directory *`$GSAF_HOME/sddf/data/load_config`*.

The <config_file>.cfg has the following contents: (without line number)

Line	Attribute Name	Possible Values
1	**ElementConfiguration**	**<Element Name>**
2	**MissionName**	**<Mission Name>**
3	**SystemTreeVersion**	**<Systemtree Version Number>**
4	**CCUPathname**	**<Pathname of CCU>**
5	**CCUName**	**< Name of CCU >**
6	**CCUVersion**	**< CCU Version Number>**
7	**CCUIssue**	**< CCU Issue Number>**
8	**CCURevision**	**< CCU Revision Number>**
9		
10	**SWDirName_Set_1**	**<Name of Directory 1>**
11	**Executable_Set_1**	**<Name of Executable 1>**
12	**Enditem_Pathname_Set_1**	**<full MDB Pathname of Entity 1>**
...	...	
n	**SWDirName_Set_m**	**<Name of Directory m>**
n+1	**Executable_Set_m**	**<Name of Executable m>**
n+2	**Enditem_Pathname_Set_m**	**<full MDB Pathname of Entity m>**

6.2.4.5 Load via Interactive Tool (I_MDB)

Using the I_MDB (Test Preparation), the user can initiate the SW Entity Dataset Loading either at CCU level or at End Item level:

Load at CCU Level:

The user has to select a CCU and to select File –> Show CCU Versions first to get the CCU version window displayed.

In order to load the correct data set, it is necessary to specify search parameters for the retrieval operation. These parameters have to be entered in the Tool Invocation window, which is called by selecting “Command –> Tools –> Load SW Datasets” in the CCU version window as shown in the following figure:.

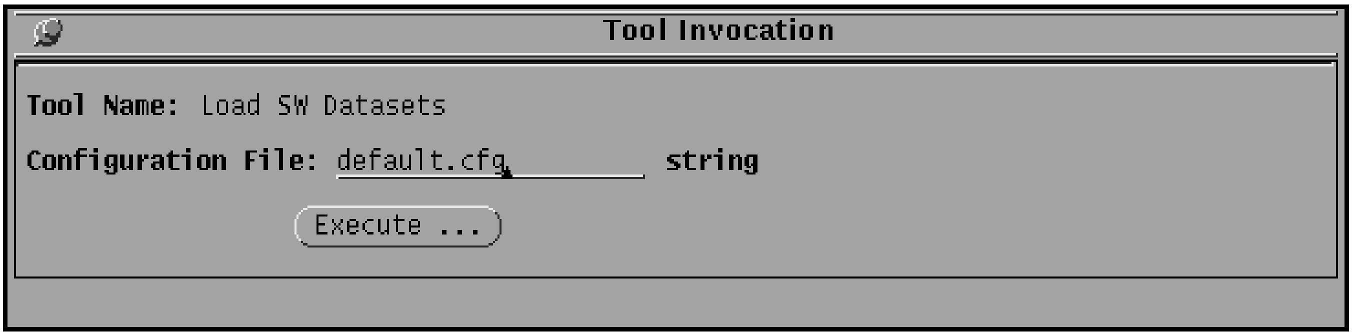


Figure 6–25 : *Tool Invocation at CCU Level*

Load at End Item Level:

The user has to navigate down to the enditem node, selecting either a CCU or a CDU version, where the SW Entity End Item is placed.

- *It is recommended that the the loading of SW Entity data sets is only performed within the scope of the CCU, which contains references to the SW Entity end item(s) and to CDU / CCU versions which contain / refer to End Items used during the pre-compilation process.*

In order to load the correct data set, it is necessary to specify search parameters for the retrieval operation. These parameters have to be entered in the Tool Invocation window, which is called by selecting “Tools –> Load SWEU Dataset” resp. “Tools –> Load SWRU Dataset” as shown in the following figure:.

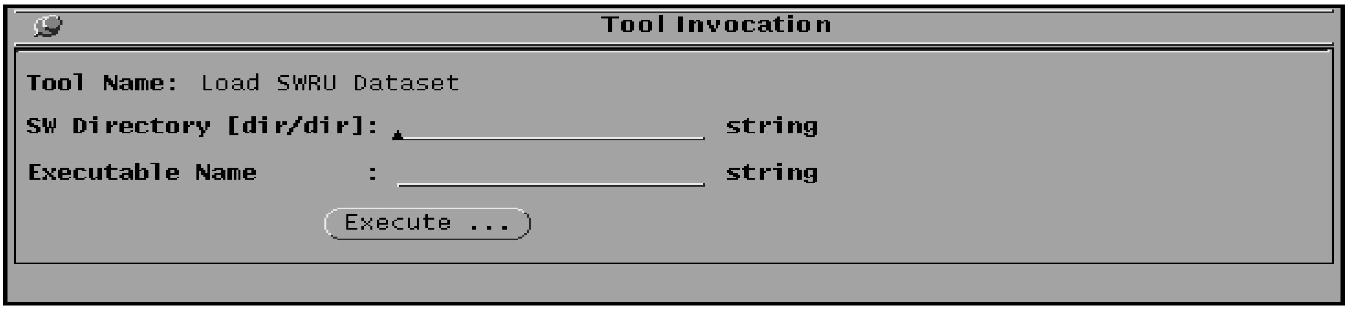


Figure 6–26 : *Tool Invocation at End Item Level*

The *Directory [dir/dir]* is to be specified relative to the *$GSAF_HOME/sddf/data/sw* directory. Under this directory the following files are expected (to be created by the CM System):

ExecutableName	executable file of SWRU resp. SWEU
ExecutableName.att	Attribute file as specified below
ExecutableName.xref	Cross reference file containing references to MDB Enditems. This file is optional. It may be omitted, if SWRU /SWEU code does not contain MDB pathname

During load operation, a progress bar will indicate the active process.

After having finished the load operation, a list of loaded files will be presented together with the scope used.

Error Reporting:

In case of errors, a message will appear in the Message Window, specifiying the name of the log file where erros are described. The error logs will be presented to the user in interactive windows.

For an example see the following figure:

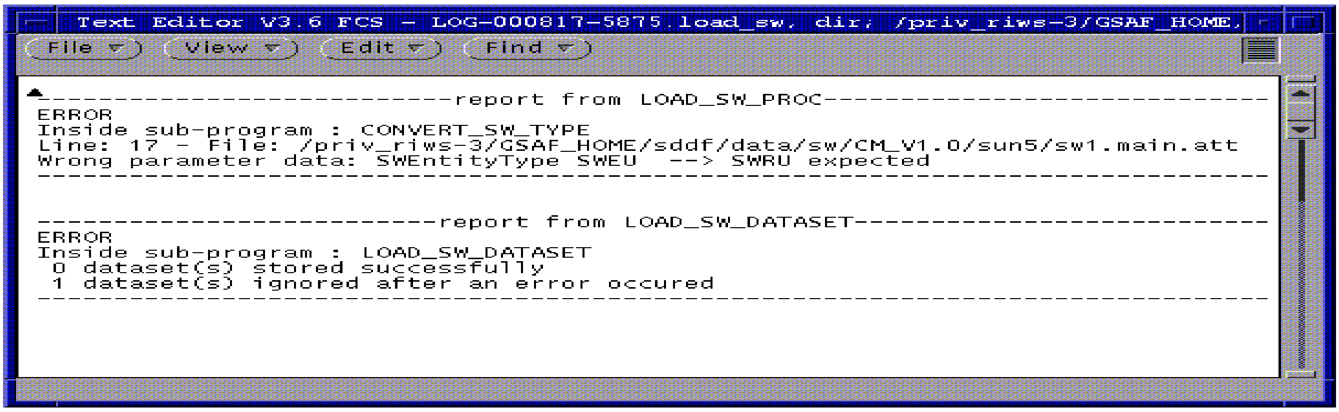


Figure 6–27 : *Load SW Entity – Error report*

6.2.5 Defining On-Board Synoptic Displays

The Flight Window Definition Utility (FWDU) is the CGS tool for preparing on–board synoptics..

The user should refer to the FWDU User Manual (see reference 2.1.2.3) for details on use of this tool.

7 TEST PREPARATION

For a more detailed information, one should also read the mission database documentation!

Before beginning with test preparation you should first familiarize yourself with chapters 6.1 and 6.2 regarding the concept and use of the MDB tool.

7.1 Creating a new MDB configuration

This chapter explains how to create a new MDB configuration before running a test. A description is provided of all data to put in the MDB which are mandatory to make a configuration running as a test.

Note that certain information is most likely already in the database and just needs to be referenced in the CCU. This applies specially for HK_VALUES, Test configuration, User definitions, Workstations and EGSE node definitions.

7.1.1 Organising the data

The structuring of data – specially the partitioning in CDUs – is used to minimize the redundant creation of data and to maximize reusage of data by referencing.

The following end items should be separated out, i.e. for each category an individual CDU should be created.

* Test execution node, database node, simulation node and participating workstation node end items

* Software Variables

* Test configuration end items

* UCL System Libraries

* Ground symbol library in case Synoptic Displays are using Symbols

* Simulation models and

* Software variables for simulation model top level inputs/outputs

The organisation of the user data should be done considering

- meaningful structuring of the tree, allowing to handle groups of data as a virtual subtree. This is especially applicable for measurements/sw_variables/derived values as well as ADUs and GDUs.
- management aspects with respect to version control and import/export handling (don't mix data from different development groups or life cycles in one CDU)
- avoidance of spreads over different CDUs where possible (group data referencing eachother together). This makes it easier to get to consistent data subsets.
- separate test data from operational data

To generate a certain test configuration a CCU has to be created which references all CDUs that are necessary to define the test configuration.

7.1.2 System prerequisites

Before you start with test preparations take a short look at the following sections. The files SYSTEM_TOPOLOGY_TABLE and USER_PROFILES contain certain restrictions. The HW and the processes running on the HW are defined here and the user rights are given.

7.1.2.1 The SYSTEM_TOPOLOGY_TABLE

The SYSTEM_TOPOLOGY_TABLE is maintained by the **CGS administrator** only.
CGS provides the script `change_system_topology` which allow several tasks to modify the system topology table.

Following features can be changed in the system topology table:

- `Add new workstation`
- `Add new test node`
- `Add processes to a known host`
- `Add SAS to a known host`
- `Remove some processes or SAS for a host`
- `Remove a host completely`
- `Remove a process completely`

To add an SAS to a known host, you have to insert the **full** SAS name (e.g. SAS_TMTC, TTS_01), and a port number.

The changed SYSTEM_TOPOLOGY_TABLE is then created automatically.

```
#
-------------------------------------------------------------
# An example SYSTEM_TOPOLOGY_TABLE is given below:
# In general, this example will not be accepted on your
# infrastructure, because all hostnames must be defined
# in your /etc/hosts file.
#------------------------------------------------------------

SITE:DASA-RI

cgs_1: HCI_01
cgs_1: TSCV_01
cgs_1: TEV_01

vicos_1: HCI_12

svf_5: HCI_09
svf_5: TEV_02

aiv_s: DBS_01
aiv_s: DBS_02
aiv_s: DBS_03

hp847: TES_01
css3s: CSS_01

eureca-1: TES_02
svf_tn: TMTC_VIEWER1:7690
```

Figure 7–1 : *The SYSTEM_TOPOLOGY_TABLE shows the assignment of machine name and internal application (logical) name*

Names used in the end items are the CGS internal application names not the UNIX name of the machine. To find the internal names of a workstation, a DB server or a test execution node you have to look into the SYSTEM_TOPOLOGY_TABLE file located in the UNIX directory $GSAF_HOME/cgs/config or use the TSCV Program.
The names on the left side of the colon are the UNIX host machine names (as laid down in the /etc/hosts file after the internet address), the names on the right side of the colon are the internal application (logical) names.

To define the end items of type EGSE_NODE correctly the user has to know the logical name of the machines used in the test.

7.1.2.2 Defining the test user profile

A User Profile is a set of information which describes user environment and controls the allowed activities of a CGS user.

The file USER_PROFILES is stored centrally in the $GSAF_HOME/cgs/config directory. The file is maintained by the **CGS administrator** only.

The following table proposes a mapping between the User Task and the required privileges.

	Test Conductor	Test Operator	Test Evaluator
Onboard SW development		**X**	
Full mission preparation		**X**	
Minimum mission preparation		**X**	
System/subsystem checkout	**X**		
System/subsystem simulation	**X**		
System/subsystem model development		**X**	
Off–line test evaluation			**X**
SAS software development		**X**	

For each CGS user one (multi–line) entry is comprised with the layout as follows:

* NAME=<UNIX user name>

* PRIV=<IS_TEST_CONDUCTOR | IS_TEST_OPERATOR | IS_TEST_EVALUATOR>

* HCI_SCREEN_SETUP=<file name>

* SERVICE=<SERVICE_NAME><SEPARATOR><UNIX_PATHNAME>

The **NAME** starts the user entry and defines its name. The name specified for the user has to be the same as the UNIX user name. This line is obviously **mandatory** for each user entry.

The **PRIV** entry determines the privileges of the user. A user can either be:
IS_TEST_CONDUCTOR : Users with this privilege can perform

– test system set-up
– online test control through HLCL commands
– view synoptic displays and predefined status displays
– perform test evaluation

IS_TEST_OPERATOR : Users with this privilege can perform
– online test control through HLCL commands
– view synoptic displays and predefined status displays
– perform test evaluation

IS_TEST_EVALUATOR : Users with this privilege can perform
– view synoptic displays and predefined status displays
– print a file
– perform test evaluation
The following is forbidden:
– HLCL commands to test nodes
– Setting values to SW Variables

Note: Users being test observers should have this privilege only

This line is **mandatory** for all users.

The entry **HCI_SCREEN_SETUP** defines the standard screen set-up to be used for the user.
All available screen set-ups are centrally stored in the directory HCI_HOME/data/screen_setup_pool. The file name must be the name of a file in this directory. This line is **optional**.

Up to eight ”private” **services** may be defined in the HCI menu. The service name is the label of the service in the menu entry. The separator is a semi-colon. The unix pathname is the full pathname of a shell or executable to be executed whenever the user activates the corresponding menu entry. This line is **optional**.

```
#--------------------------------------------------------------
# An example USER_PROFILES is given below:
#--------------------------------------------------------------

NAME=cgsadmin
PRIV=IS_TEST_CONDUCTOR
HCI_SCREEN_SETUP=cgsadmin
SERVICE=Message Window; /gsaf_home/cgsi/bin/sun4/Start.Message_Window
SERVICE=Test Setup; /gsaf_home/tscv/bin/common/start_tscv
SERVICE=Mission Data Base; /gsaf_home/mda/bin/common/I_MDB
SERVICE=Shelltool; shelltool -Wl cgsadmin

NAME=cgsadmin
PRIV=IS_TEST_OPERATOR
HCI_SCREEN_SETUP=grotheer
SERVICE=Message Window; /gsaf_home/cgsi/bin/sun4/Start.Message_Window
SERVICE=Mission Data Base; /gsaf_home/mda/bin/common/I_MDB

NAME=cgsadmin
PRIV=IS_TEST_EVALUATOR
SERVICE=Message Window; /gsaf_home/cgsi/bin/sun4/Start.Message_Window
SERVICE=Evaluate; /gsaf_home/tev/bin/common/start_tev

NAME=cgsadmin
PRIV=IS_TEST_OPERATOR
HCI_SCREEN_SETUP=svf_operator
SERVICE=Message Window; /gsaf_home/cgsi/bin/sun4/Start.Message_Window
SERVICE=Mission Data Base; /gsaf_home/mda/bin/common/I_MDB

NAME=cgs_1
PRIV=IS_TEST_CONDUCTOR
HCI_SCREEN_SETUP=cgs_1
SERVICE=Message Window; /gsaf_home/cgsi/bin/sun4/Start.Message_Window
SERVICE=Test Setup; /gsaf_home/tscv/bin/common/start_tscv
SERVICE=Mission Data Base; /gsaf_home/mda/bin/common/I_MDB
SERVICE=Shelltool; shelltool -Wl cgsadmin

END_USER_PROFILES
```

Figure 7–2 : *The file USER_PROFILES defines the individual privileges and defaults*

7.1.3 Creating a new MDB configuration – Defining DB end items

There are a number of database enditems which describe the test facility and the configuration of the facility for a given test. This includes definitions of the individual HW nodes of the test equipment, the individual SW entities and the overall configuration.

7.1.3.1 Defining the node list

Each test configuration has to contain at least three standard node types: a Workstation node, a DB server node and a Test node. These nodes reside on – at least – two computers: one executing the Test node and the other executing the Workstation node and the DB Server node.
As soon as the test configuration is expanded e.g. a Simulation node running CSS models, it is necessary to expand the configuration with more computers.

The node list itself serves as an input list for the definition of test configurations.

The node list consists of end items of type EGSE_NODE.
These enditems describe a specific part of the HW of the facility. The main purpose is to provide access to parts of the facility via pathnames from UCL/HLCL level, but the end item also defines the 'role' of the node, e.g. in the EGSE environment it determines whether the node is a workstation, a test node, a database server...

To define an end item of type EGSE_NODE three inputs can be made:

- Node Type – this determines wether the node is a is a workstation, a test node, a database server, a simulation node, a front–end equipment or a UUT.
- Logical Name – this connects the desired HW (see section 7.1.2.1)
- CGS Internal Name – this is an internal marker for the checkout and test system

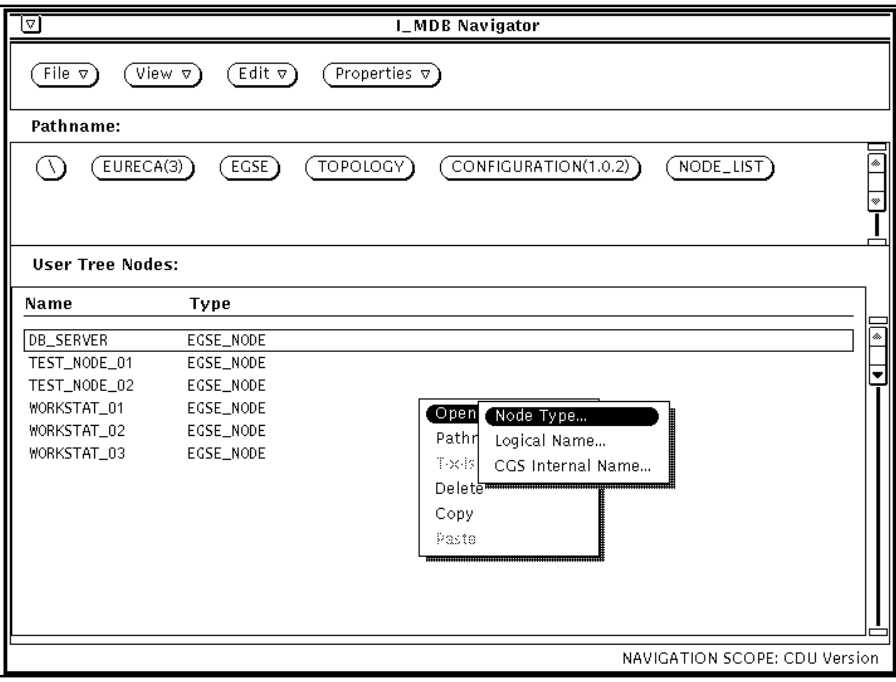


Figure 7–3 : *Each entry in the node list can be defined by three attributes – Node Type, Logical name and CGS internal name.*

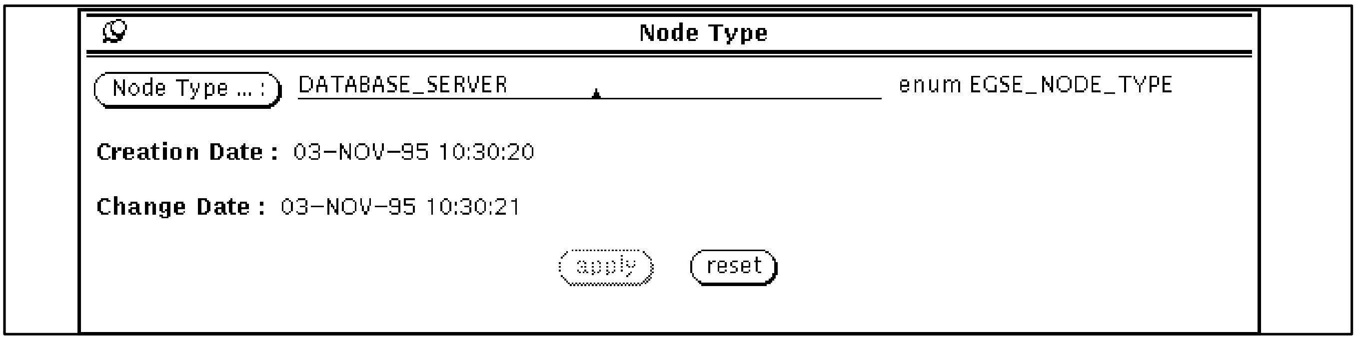


Figure 7–4 : *The Node Type defines the function of the node in the test configuration*

For the node types WORKSTATION, SIMULATOR, DATABASE_SERVER and TEST_NODE following rules are valid:

- all three inputs are mandatory
- the logical name is the machine name which can be found in the SYSTEM_TOPO-LOGY_TABLE
- a workstation node has the internal prefix HCI
  a simulator node has the internal prefix CSS
  a test node has the internal prefix TES
  a database server has the internal prefix DBS

Note that you can't choose any node type you like.
The assignment (which task may be performed on which computer) is already made in the SYSTEM_TOPOLOGY_TABLE.
Example: Take a look at Figure 7–1. The computer svf_5 *may be used as workstation, the machine* hp847 *as a test node, the* aiv_s *as database server. Note that in the last case it is possible to define three different DB nodes.*

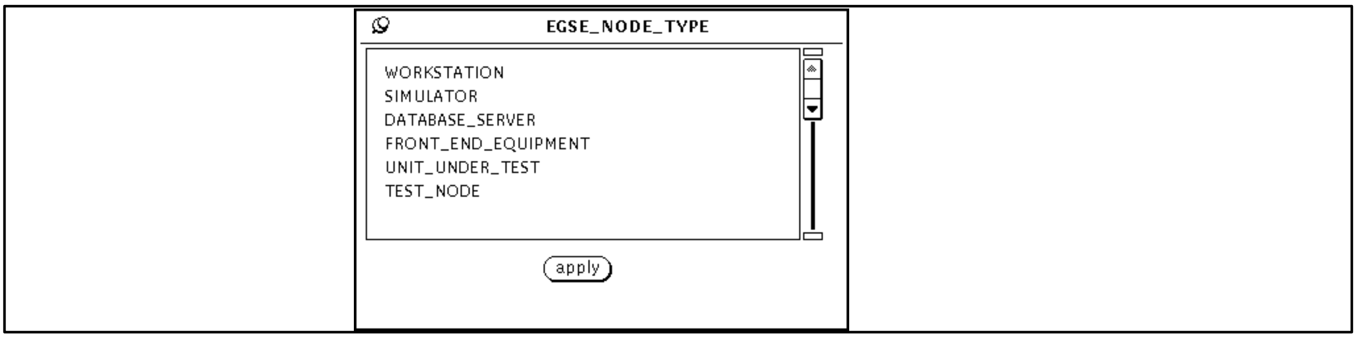


Figure 7–5 : *The node type describes the function of the node in the test configuration*

For the node types UNIT_UNDER_TEST and FRONT_END_EQUIPMENT the prefix input should be omitted. The logical name depends on the type of HW, i.e. the logical name could be an internet address, a bus address etc.

The following procedure is valid for the four common node types used in a test configuration.

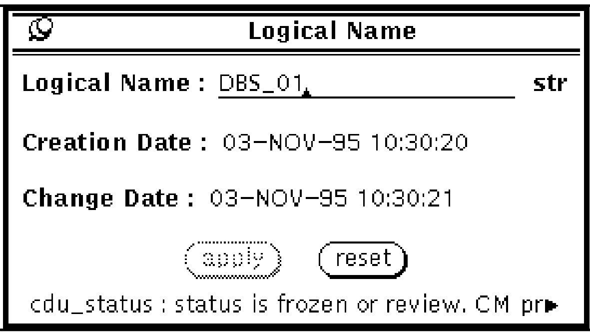


Figure 7–6 : *The computer with the logical name DBS_01 is linked to the node definition*

Defining a DB server node

- If not already done, create a CDU (domain: EGSE) for the node definitions.
- *Note that a node is not necessarily a separate computer system !*
- For each node repeat the following steps.
- Create an end item of type EGSE_NODE.
- Select the end item.
- Press the right mouse button and select **Open –> Node Type...**
- The Node Type window appears (see Figure 7–4) Press the **Node Type...** button.
- A selection list window pops up. Select the node type DATABASE_SERVER.
- **Apply** your input.
- Press the right mouse button and select **Open –> Logical Name...**
- Type the logical name of the desired computer in the **Logical Name:** field (see Figure 7–6).
- Press the **apply** button.
- Press the right mouse button and select **Open –> CGS Internal Name...**
- The CGS Internal Name window appears. Press the **CGS Prefix...** button.
- A selection list window pops up. Select the CGS_Node_Prefix type DBS.
- **Apply** your input.

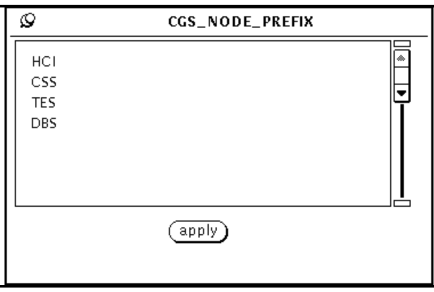


Figure 7–7 : *The prefix is an internal marker*

The node list is later used as a selection list for the definition of a test configuration.

7.1.3.2 Defining the test configuration

For each test configuration an end item of type EGSE_TEST_CONFIGURATION has to be created.

This enditem describes the actual configuration of the facility for a given application. For the checkout and test system, this is the actual configuration of the test equipment to be set–up for a given test.

The test configuration is described in terms of references to other MDB items of type EGSE_NODE and EGSE_SW plus the definition of MDB contents to be ’downloaded’ to test nodes.

Enditems of type test configuration are later used in the CGS product TSCV to actually do set-up.

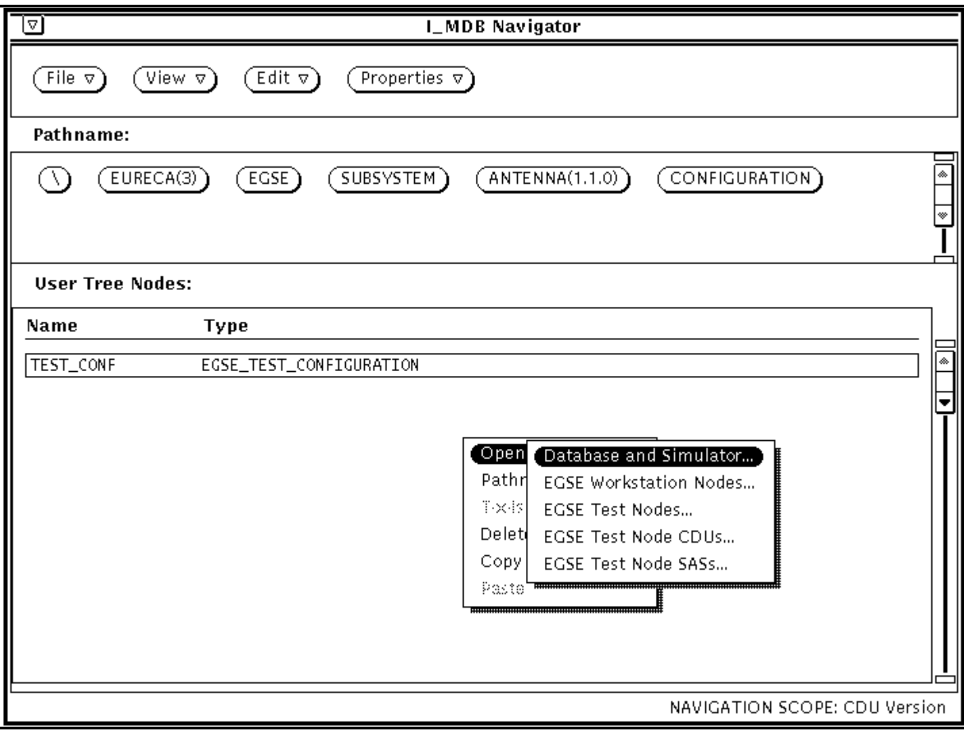


Figure 7–8 : *To describe a test configuration all information must be supplied here*

Note that all the following procedures are needed to define a test configuration.
Instead of writing one large procedure the procedure has been split into several smaller ones to increase readability and comprehensibility.

Defining the test configuration – Database and Simulator node

- If not already done, create a CDU (domain: EGSE) for the test configuration definitions.
- For each test configuration repeat the following steps.
- ■ Create an end item of type EGSE_TEST_CONFIGURATION.
- ■ Select **Open–>Database and Simulator** and an input window appears.
- ■ Press the **Database Node ...:** button. An **Item Reference Help** window pops up.

to be continued ...

- Press the **Select CDU versions...** button. A CDU Version Selections window appears displaying a list of available CDU version.
- Select the CDU version where the database node is located. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the database node and press the **Apply** button.
- The complete DB path including the database node end item is displayed in the corresponding input line (see Figure 7–9). Press the **apply** button again.

The definition of a simulation node is not mandatory.

- Press the **Simulation Node ...:** button. An **Item Reference Help** window pops up.
- Press the **Select CDU versions...** button. A CDU Version Selections window appears displaying a list of available CDU version.
- Select the CDU version where the simulation node is located. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the simulation-node and press the **Apply** button.
- The complete DB path including the simulation node end item is displayed in the corresponding input line. Press the **apply** button again.

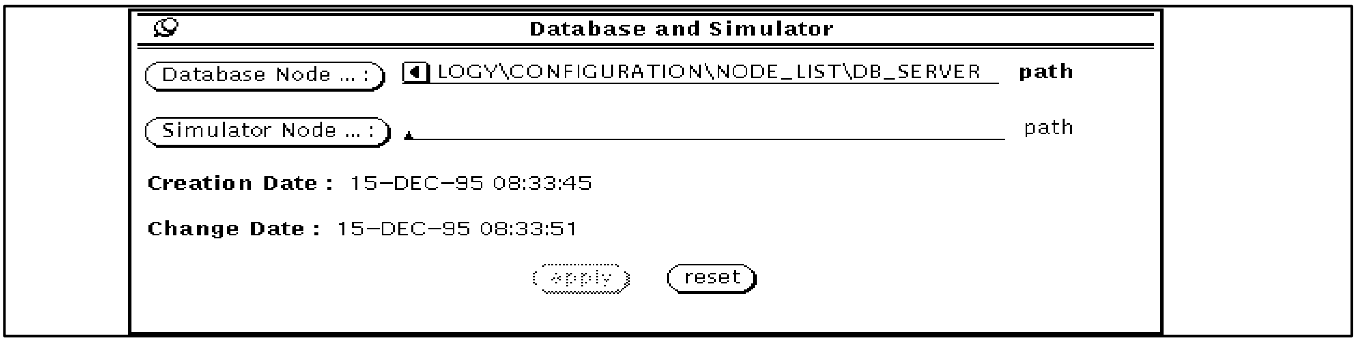


Figure 7–9 : *The database and simulator nodes are identified by their pathname*

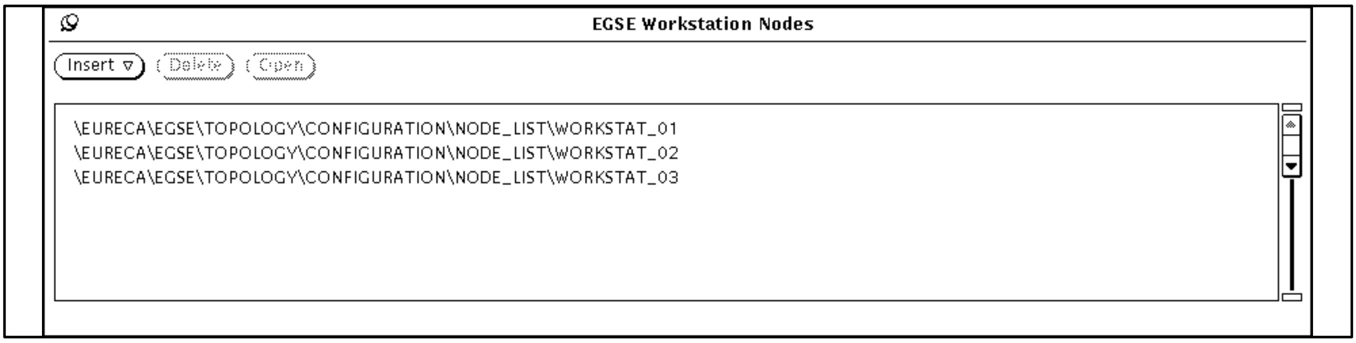


Figure 7–10 : *The workstation node list*

Defining the test configuration – Workstation nodes

- If not already done, create a CDU (domain: EGSE) for the test configuration definitions.
- For each test configuration repeat the following steps.
- If not already done, create an end item of type EGSE_TEST_CONFIGURATION.
- *Repeat the following steps for each participating workstation node.*
- Select the test configuration end item.
- Select **Open–>EGSE Workstation Nodes...** and an **EGSE Workstation Nodes** list window pops up (see Figure 7–10).
- Press the **Insert:** button. An **EGSE Workstation Nodes** definition window pops up.
- Press the **Workstation Node...:** button. An **Item Reference Help** window pops up.
- Press the **Select CDU version...** button. A **CDU Version Selection** window appears displaying a list of available CDU versions.
- Select the CDU version where the workstation node is defined. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the desired workstation node and press the **Apply** button.
- Press the **Is Participating** button, select TRUE and press the **apply** button
- The complete DB path including the workstation node end item is displayed in the corresponding input line (see Figure 7–11). The **Is Participating** line shows TRUE. Press the **apply** button again.
- Close the window by clicking on the push pin.

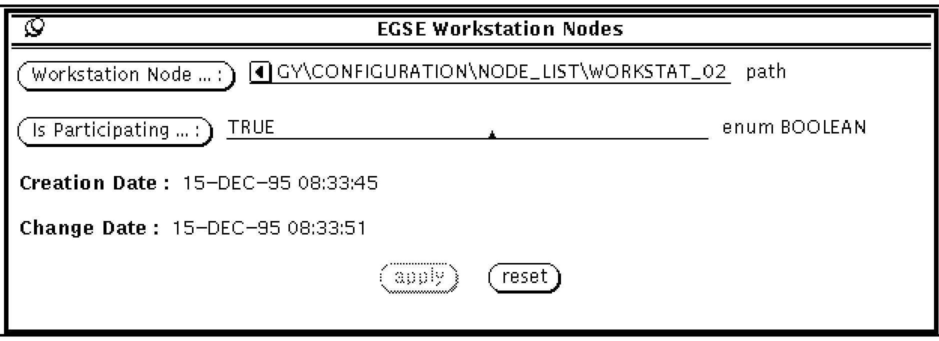


Figure 7–11 : *The workstation node is participating*

In a test configuration there is at least one test node required.

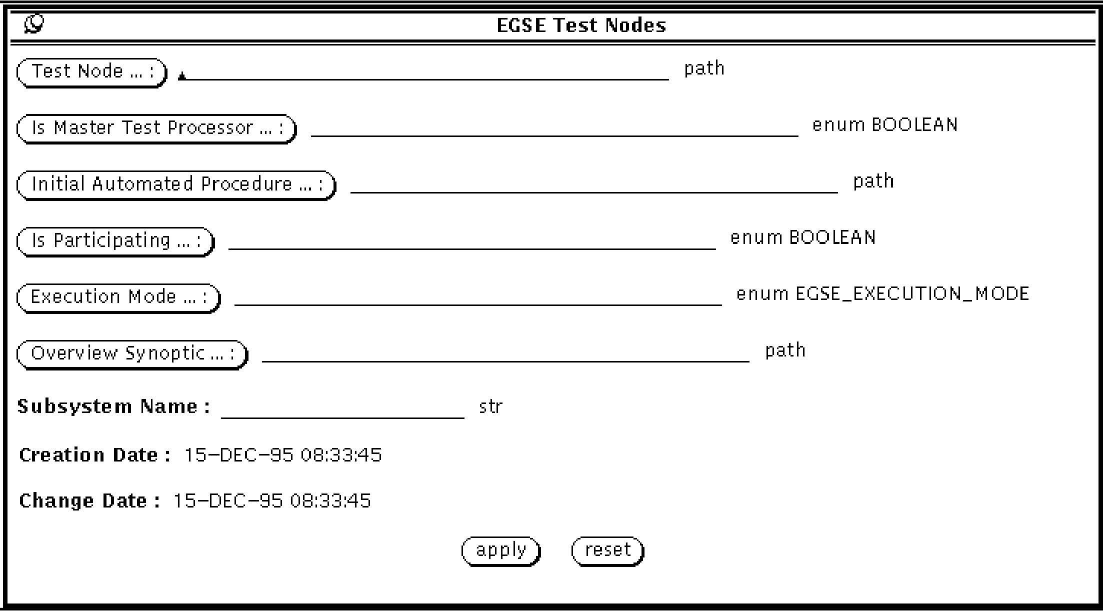


Figure 7–12 : *To create a testnode fill in the test node definition window*

Defining the test configuration – Test nodes

- If not already done, create a CDU (domain: EGSE) for the test configuration definitions.
- For each test configuration repeat the following steps.
- If not already done, create an end item of type EGSE_TEST_CONFIGURATION.

Repeat the following steps for each participating test node.

- Select the TEST_CONFIGURATION end item.
- Select **Open–>EGSE Test Nodes...** .
- Press the **Insert...** button and an **EGSE Test Nodes** definition window pops up (see Figure 7–12).
- Press the **Test Node...:** button. An **Item Reference Help** window pops up.
- Press the **Select CDU version...** button. A **CDU Version Selection** window appears displaying a list of available CDU versions.
- Select the CDU version where the test node is defined. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the desired test node and press the **Apply** button.

If there is only one test node in the configuration this must be the MTP (Master Test Processor)
The definition of a MTP is mandatory for two reasons:
– setting the SMT (Simulated Mission Time) is done on the MTP only
– there is a selection of HK (House Keeping) data only available from the MTP

- Press the **Is Master Test Processor...:** button. An **input** window pops up.
- Select TRUE and press the **apply** button.

to be continued ......

- Perform the following steps for each additional participating test node.
- Press the **Is Master Test Processor...:** button. An **input** window pops up.
- Select FALSE and press the **apply** button.

The following input is not mandatory.
With the initial AP defaults can be set for a test node and an application can be started on the desired test node automatically.

- Press the **Initial Automated Procedure...:** button. An **Item Reference Help** window pops up.
- Press the **Select CDU version...** button. A **CDU Version Selection** window appears displaying a list of available CDU versions.
- Select the CDU version where APs are defined. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the desired AP and press the **Apply** button.
- Press the **Is Participating ... :** button, select TRUE and press the **apply** button

There are three execution modes: NORMAL, REPLAY and SIMULATION
In NORMAL mode the tests with the UUT are performed, in REPLAY mode the user will see the operations of a previously executed test, in SIMULATION mode all incoming data is generated within the test nodes itself (refer to section 3.2.6 for more information).

- the **Normal** test execution mode
  in this mode the nominal tests with the UUT are performed.
- the **Replay** mode
  in this mode the user will see the operations of a previously executed test. The replay submode shall be used to look to events anticipated by the operator which were not encountered during tests or to investigate situations before test deviations occurred. The data presentation will be in the same way as for normal mode, but all data that are generated to interfaces external to the checkout and test system are suppressed. The replay mode may be set up with different parameters w.r.t. to timing behaviour.
- the **Simulation** test execution mode
  in this mode all incoming data is generated within the test nodes itself. The simulation of this data is driven by predefined data tables. Simulation mode is normally used for database verification, i.e. test of synoptic displays etc.

- Press the **Execution Mode... :** button, select NORMAL and press the **apply** button
- The complete DB path including the workstation node end item is displayed in the corresponding input line (see Figure 7–13). The **Is Participating** line shows TRUE. Press the **apply** button again.

The following input is not mandatory.
With the overview synoptic a default screen layout can be determined.

- Press the **Overview Synoptic... :** button. An **Item Reference Help** window pops up.
- Press the **Select CDU version...** button. A **CDU Version Selection** window appears displaying a list of available CDU versions.

to be continued .....

- Select the CDU version where APs are defined. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the desired synoptic display and press the **Apply** button.
- *The following input is not mandatory. It is made for documentation purposes.*
- Type the subsystem name in the **Subsystem Name:** line.
- Press the **Apply** button.
- Close the window by clicking on the push pin.

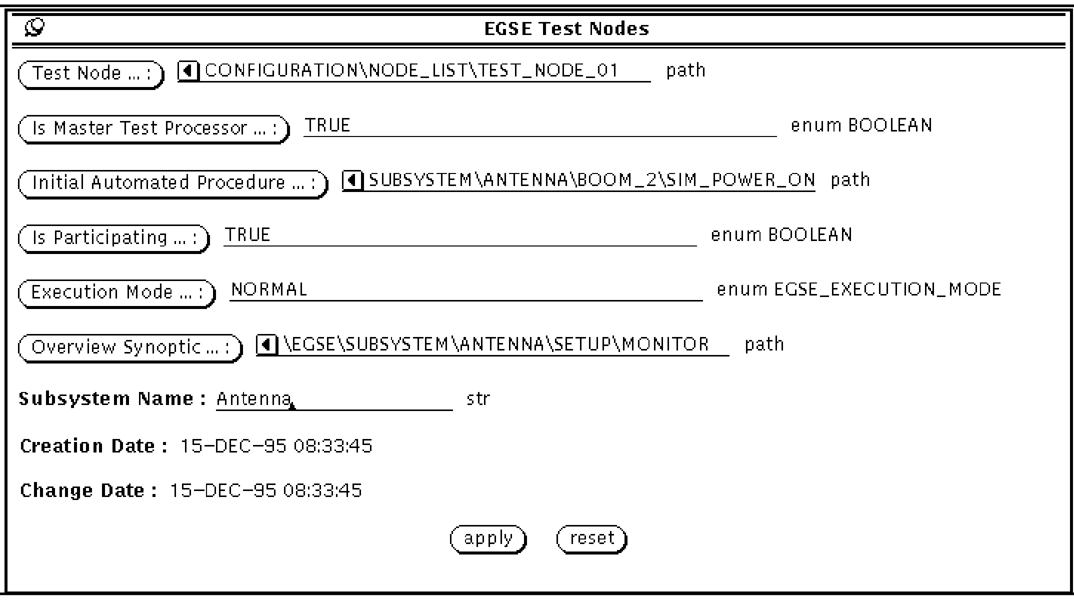


Figure 7–13 : *The completed Test Nodes window.*

It takes two steps to make data available on a test execution node:

* it has to be visible in the used CCU

* it has to be loaded to this execution node

Precondition for the execution of step two is the execution of the following procedure.
All CDUs which contain data needed for the test must be assigned to the test nodes. This includes CDUs with HK data, the configuration definition itself, displays, SASs, simulation models and SW variables, GDU description lists and APs, CCSDS Adu description and SW variables.

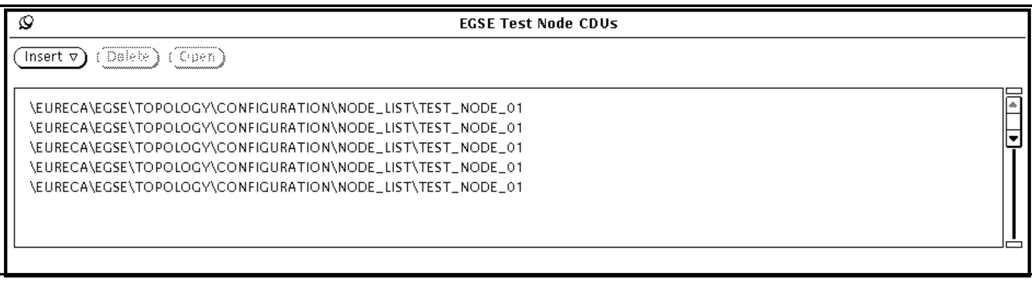


Figure 7–14 : *CDU list – five CDUs are assigned to test node TEST_NODE_01*

Defining the test configuration – Test node CDUs

- If not already done, create a CDU (domain: EGSE) for the test configuration definitions.
- For each test configuration repeat the following steps.
- If not already done, create an end item of type EGSE_TEST_CONFIGURATION.
- Select the TEST_CONFIGURATION end item.
- Select **Open–>EGSE Test Node CDUs...** and an **EGSE Test Node CDUs** list window pops up (see Figure 7–14).

For each CDU which contains things connected to the test node a reference has to be set.

- Press the **Insert:** button. An **EGSE Test Node CDUs** definition window pops up.
- Press the **Test Node...:** button. An **Item Reference Help** window pops up.
- Press the **Select CDU version...** button. A **CDU Version Selection** window appears displaying a list of available CDU versions.
- Select the CDU version where the test node is defined. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the desired test node and press the **Apply** button.

Note that it is possible to select a CDU which includes other CDUs.

- Press the **Loaded CDU...** button. An **Item Reference Help** window pops up.
- Press the **Select CDU version...** button. A **CDU Version Selection** window appears displaying a list of available CDU versions.
- Select the CDU version with the desired contents. Then press the **Apply** button.
- The **Item Reference Help** window shows now the selected CDU. Press the **Apply** button.
- The complete DB path including the test node end item is displayed in the corresponding input line (see Figure 7–15). The **Loaded CDU...** line shows the DB path of the selected CDU.
- Repeat the last five steps for all CDUs needed on that test node.
- Close the window by clicking on the push pin.

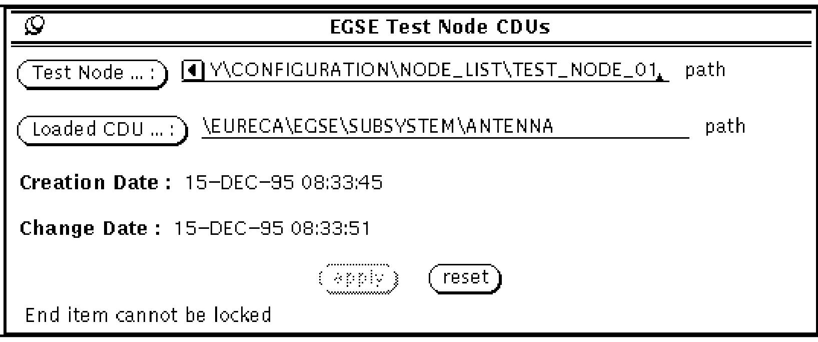


Figure 7–15 : *The CDU contents is available on TEST_NODE_01*

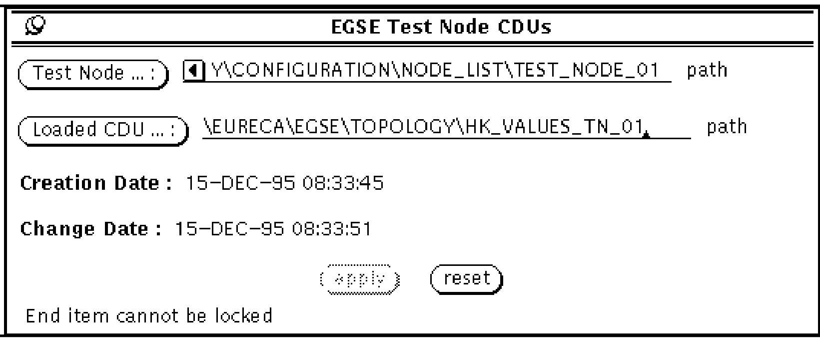


Figure 7–16 : *The house keeping values defined for TEST_NODE_01 are assigned to the test node*

Defining the test configuration – Test node SASs

- If not already done, create a CDU (domain: EGSE) for the test configuration definitions.
- For each test configuration repeat the following steps.
- If not already done, create an end item of type EGSE_TEST_CONFIGURATION.

Repeat this steps for all test nodes running SASs.

- Select **Open–>Test Node SASs.**
- Press the **Insert:** button. An **EGSE Test Node SASs** definition window pops up.
- Press the **Test Node...:** button and select the desired test node as already described.
- Press the **SAS...** button.
- The **Item Reference Help** window pops up.
- Press the **Select CDU version...** button. A **CDU Version Selection** window appears displaying a list of available CDU versions. Select a CDU and press the **Apply** button.
- The **Item Reference Help** window shows now a list of all SASa in that CDU. Select the desired SAS and press the **Apply** button.
- The complete DB paths including the test node end item and the SAS end item are displayed in the corresponding input lines (see Figure 7–11).

Note that the SAS has to be assigned to the test node in the SYSTEM_TOPOLOGY_TABLE. The database entry repeats the assignment already made in the table.

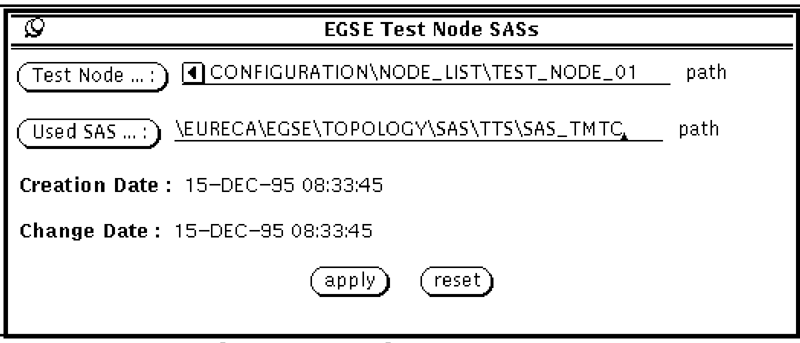


Figure 7–17 : *The SAS is assigned to a test node*

7.1.4 Defining House Keeping (HK) values

In section 7.1.12 a list of TES housekeeping data is provided.

There are no mandatory house keeping variables. Software variables are only needed for those TES housekeeping data that shall be available for APs and SASs or when values shall be displayed in synoptics.

We recommend to separate the housekeeping data into different CDUs, one for the Master Test Processor (MTP) house keeping values and one for each test node. The MTP CDU contains those HK values which are used for global overall monitoring.

Note that this variables should be defined just once per system tree version and being referenced in the different CDUs.

A software variable is to be created in the data base with a reference to the HK value. The housekeeping identifier is the number which can be found in the list. The data type must comply with the data type listed in section 7.1.12.

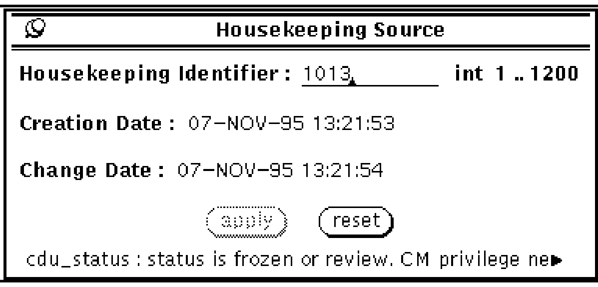


Figure 7–18 : *The housekeeping identifier is a number from the list of HK data*

7.1.5 Defining UCL Libraries

7.1.5.1 Defining UCL System Libraries for VICOS / TES

The baselined version of the CGS system libraries can be found in your installation:
Under $CGS_HOME/config the following files exist:

ground_library_.ucl	This library is needed always to control VICOS/TES (all projects)
cpl_library_.ucl	contains the library used for CPL (Columbus)
fwdu_library_.ucl	contains the library needed for the FWDU (Columbus)
ground_commands_to_onboard_.ucl	contains the library needed for APM commands (Columbus)
tcl_library_.ucl	a library specific for the SVF project (NASA SVF)

Your configuration has to contain at least one end item containing the UCL specification of the ”Ground Library”. Create in your CCU/CDU an end item of type UCL_SYSTEM LIBRARY and open it with the CLS Editor. Load the contents of the file ”ground_library_.ucl” into the editor. Compile the library and store it.

The other libraries are optional. If you need them, depends on your configuration.

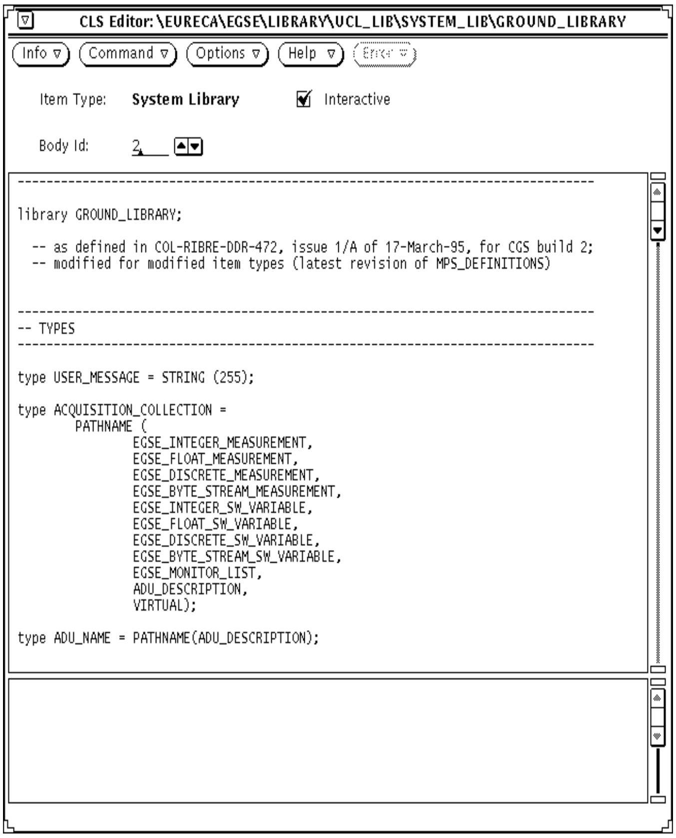


Figure 7–19 : *Ground Library loaded into CLS Editor*

Figure 7–19 shows how the CLS Editor has to be set up before compiling the Ground Library. The ”Body Id” has to be set up according to the library definition in TES. For the Ground Library, the body id is 2.

7.1.5.2 Defining UCL User Libraries

The baselined version of the CGS some user libraries can be found in your installation:
Under $CGS_HOME/config the following files exist:

file_io_lib_.ucl	FILE_IO_LIB User Library Specification. This library provides acces to UNIX files from UCL APs
file_io_lib.ucl	Body of the FILE_IO_LIB library

To install the library , create in your CCU/CDU an end item of type UCL_USER LIBRARY and open it with the CLS Editor. Load the contents of the file ”file_io_lib_.ucl” into the editor. Switch to ”BODY” and load the contents of the file ”file_io_lib.ucl” into the editor. Compile the library and store it.

Further libraries can be developped by defineing new enditems of type UCL_USER LIBRARY and creating your own UCL code.

7.1.6 Defining the connection to a model

If a CSS model (which replaces a sub–system) should be used in the test, the measurements and stimuli which connect the modelled sub–system with the rest of the system must be defined in the database. The MIL bus addresses are defined by the subsystem engineer (HW!). These addresses are then linked to corresponding individual measurements or stimuli in the database.

7.1.7 Defining the user specific configuration

In the I_MDB end items for all measurements, commands, ADUs, APs or Synoptic Displays used during the test have to be created and the current allocation must be made. Note that APs have to be compiled before using. For this purpose you need to establish a development CCU to make the right environment visible.

7.1.8 Defining SASs

Before starting the implementation of an SAS, a proper environment set–up has to be ensured in terms of setting–up

* the UNIX environment
* the compiler system environment
* the SYSTEM_TOPOLOGY_TABLE

For the first two steps contact your responsible system administrator.

As can be seen in Figure 7–1 the SASs are assigned to a host machine and to a dedicated port number. The port number for SAS has to be between 7600 and 7700.
For each SAS an end item of type EGSE_SW , which defines the SAS, has to be created.
The application name for SASs have to correspond exactly to the short names defined for the SAS in the MDB. For a detailed procedure how to create SASs refer to section 7.2.

7.1.9 Building a CCU

It takes two steps to make the information available on a test execution node:

* it has to be visible in the used CCU
* it has to be loaded to this execution node

The precondition for step 2 is the execution of procedure ”**Defining the test configuration – Test node CDUs**” described in section 7.1.3.2 .
How to create a CCU is described in detail in section 6.1.2.3.
Create references to all CDUs that are necessary to define the test configuration (see Figure 7–20)

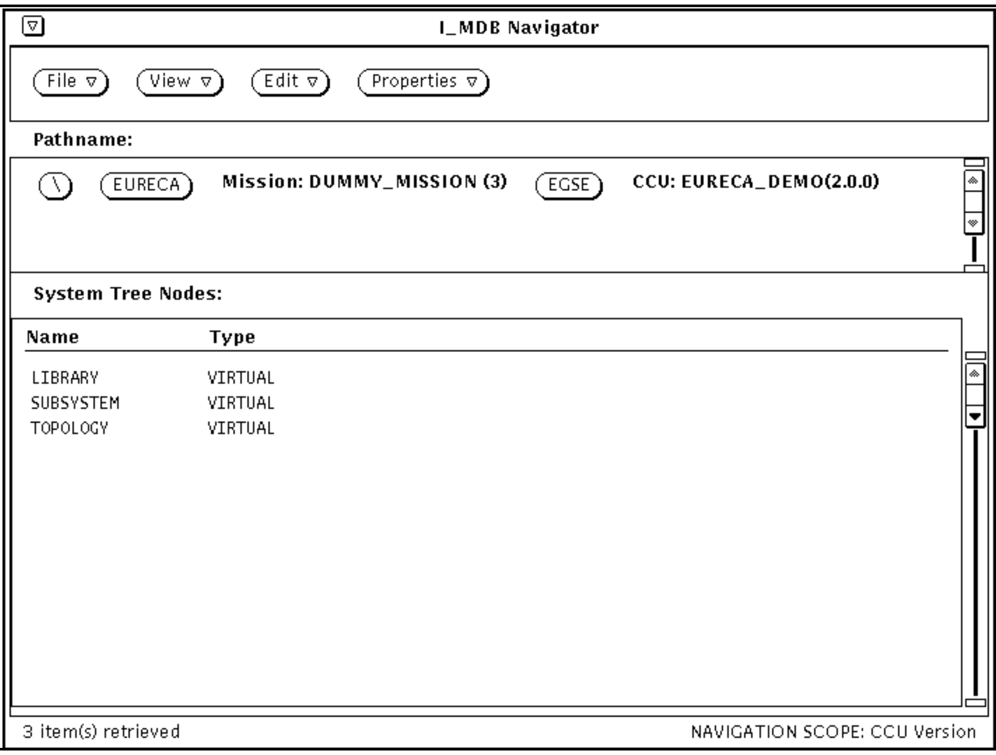


Figure 7–20 : *The CCU EURECA_DEMO references all CDUs which contents is needed for the test*

7.1.10 Performing consistency checks

Execute the consistency checker on the new/changed CCU and eliminate the marked problems. Specially all reference errors have to be solved before generating the Load Scoe file.

7.1.11 Generating the Scoe load file

Note that the CCU has to be selected before the following steps can be performed.

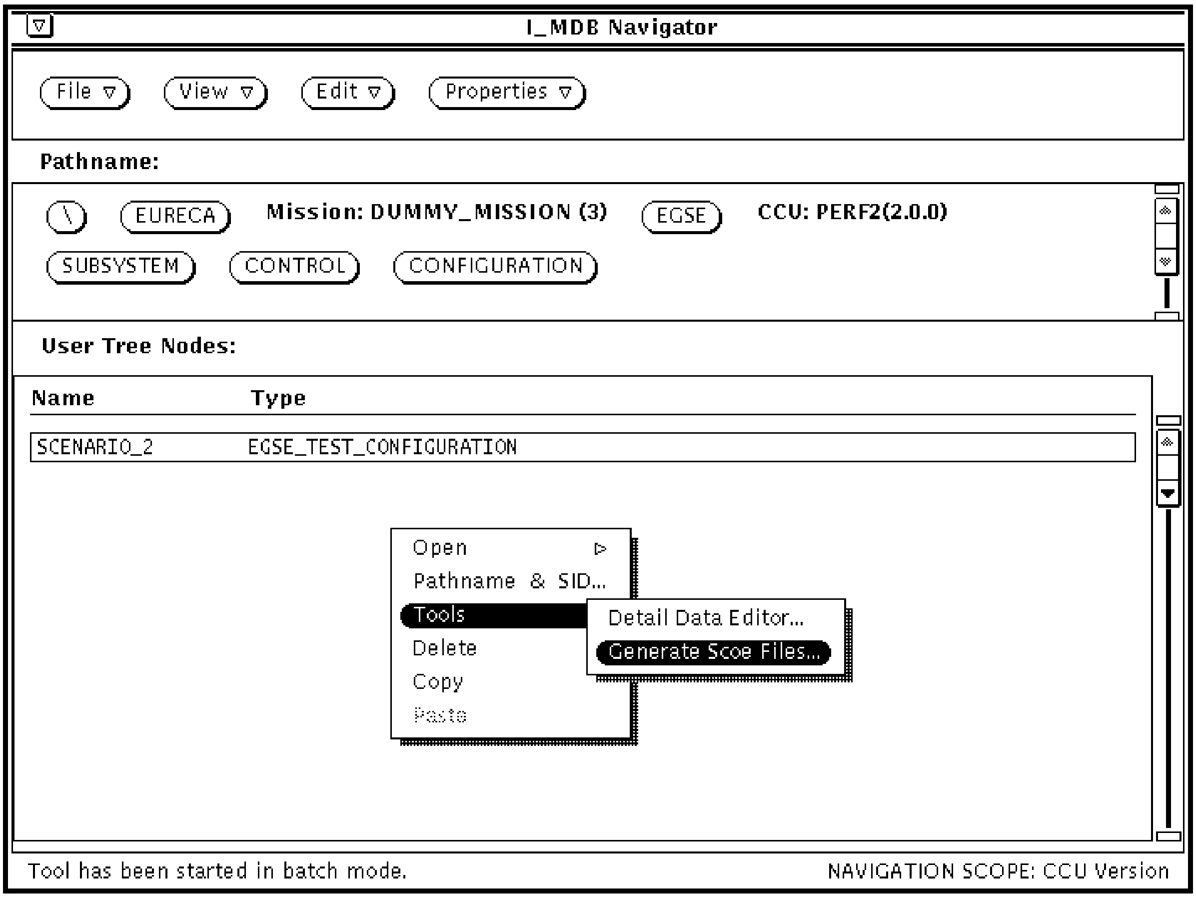


Figure 7–21 : *I_MDB Navigator provides the option to start the Scoe file generation process*

Generate the Scoe file for each EGSE Test Configuration you may want to select via TSCV lateron in this CCU by selecting **Tools–>Generate Scoe Files** in the corresponding EGSE_TEST_CONFIGURATION end item menu for this node (Figure 7–21). Look in the Console window for generation messages:

```
 Program  /sivq_gsafhome/mda/bin/sun5/start_load_scoe
 Version 3.1.X a, (C) Daimler–Benz Aerospace AG, RI Bremen
 called with parameters :
   1 :  EURECA
   2 :  DUMMY_MISSION
   3 :  3
   4 :  CCU
   5 :  \EURECA\EGSE
   6 :  PERF2
   7 :  2
   8 :  0
   9 :  0
  10 :  \EURECA\EGSE\SUBSYSTEM\CONTROL\CONFIGURATION\SCENARIO_2
  11 :  EGSE_TEST_CONFIGURATION

****  Loaded and packed tree with 1531  nodes  ****

****  Printed tree in $MDA_HOME/data/test/load_scoe.tree  ****


****  Printed test configuration in $MDA_HOME/data/test/load_scoe.tcfg  ****


****  Printed data for node TES_01  on file $MDA_HOME/data/test/
load_scoe_TES_01.lst ****


****  Printed data for node TES_02  on file $MDA_HOME/data/test/
load_scoe_TES_02.lst ****


****  Program successfully terminated  ****
```

1

2

3

Figure 7–22 : *Output of start_load_scoe in console window*

The first block of messages (Figure 7–22, section 1) describes the version of the ”Start Load Scoe” program and the parameters passed to it. Then the following output (Figure 7–22, section 2) lists the names of the files generated during the Scoe file generation process. These files are generated mainly for debugging purposes. In the example above these are:

`$MDA_HOME/data/test/load_scoe.tree:` Contains the name tree data as written to the Scoe file.

`$MDA_HOME/data/test/load_scoe.tcfg:` Contains the test configuration data as written to the Scoe file.

`$MDA_HOME/data/test/load_scoe_TES_01.lst:` Contains the listing for the data for the first test node (in this case TES_01).

`$MDA_HOME/data/test/load_scoe_TES_02.lst:` Contains the listing for the data for the second test node (in this case TES_02).

The generation of the Scoe file was only successful when the message ”Program successfully terminated” is displayed (Figure 7–22, section 3).

7.1.12 List of available HK DATA

Id	Name	Purpose	Type	Initial Value
1	”Init_Path”	”Pathname used for LOAD_SCOE”	STRING_TYPE	””
2	”TES_Mode”	”Current Mode (NONE, NORMAL, REPLAY, SIMULATE)”	STATE_CODE_TYPE	$NONE
3	”TN_Status”	”TN Status (IDLE, RUNNING, SUSPEND, ERROR)”	STATE_CODE_TYPE	$IDLE
4	”MTP_Mode”	”The value for the MTP mode (SCOE, MTP)”	STATE_CODE_TYPE	$SCOE
5	”CCU internal version”	MDA internal identifier of the CCU version used	INTEGER_TYPE	0
6	”Replay_Speed”	”The replay speed”	INTEGER_TYPE	100
7	”Archive_Time”	”The cycle time (min) an archive file will be closed”	INTEGER_TYPE	0
8	”Archive_State”	”The current state of archiving (DISABLED, ENABLED)”	STATE_CODE_TYPE	$DISABLED
10	”Local_Time”	”Actual Local Clock Value”	STRING_TYPE	””
11	”SMT”	”Actual Simulated Mission Time”	STRING_TYPE	””
12	”RLT”	”Recorded Local Time as read from archived data”	STRING_TYPE	””
20	”Active_APs”	”Number of started (but not terminated) APs”	INTEGER_TYPE	0
21	”Susp._APs”	”Number of APs explicitely suspended”	INTEGER_TYPE	0
30	”RPL_Begin_Time”	”Replay begin time selected during init”	STRING_TYPE	””
31	”RPL_End_Time”	”Replay end time selected during init”	STRING_TYPE	””
90	”Free_Disk”	”Number of free bytes on local disk”	INTEGER_TYPE	0

Id	Name	Purpose	Type	Initial Value
100	”SAS1_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
101	”SAS1_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
102	”SAS1_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
103	”SAS1_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
104	”SAS1_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
105	”SAS2_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
106	”SAS2_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
107	”SAS2_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
108	”SAS2_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
109	”SAS2_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
110	”SAS3_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
111	”SAS3_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
112	”SAS3_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
113	”SAS3_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
114	”SAS3_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
115	”SAS4_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
116	”SAS4_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
117	”SAS4_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
118	”SAS4_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
119	”SAS4_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0

Id	Name	Purpose	Type	Initial Value
120	”SAS5_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
121	”SAS5_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
122	”SAS5_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
123	”SAS5_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
124	”SAS5_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
125	”SAS6_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
126	”SAS6_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
127	”SAS6_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
128	”SAS6_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
129	”SAS6_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
130	”SAS7_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
131	”SAS7_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
132	”SAS7_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
133	”SAS7_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
134	”SAS7_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
135	”SAS8_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
136	”SAS8_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
137	”SAS8_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
138	”SAS8_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
139	”SAS8_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0

Id	Name	Purpose	Type	Initial Value
140	”SAS9_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
141	”SAS9_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
142	”SAS9_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
143	”SAS9_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
144	”SAS9_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
145	”SAS10_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
146	”SAS10_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
147	”SAS10_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
148	”SAS10_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
149	”SAS10_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
150	”SAS11_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
151	”SAS11_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
152	”SAS11_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
153	”SAS11_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
154	”SAS11_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
155	”SAS12_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
156	”SAS12_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
157	”SAS12_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
158	”SAS12_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
159	”SAS12_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0

Id	Name	Purpose	Type	Initial Value
160	”SAS13_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
161	”SAS13_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
162	”SAS13_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
163	”SAS13_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
164	”SAS13_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
165	”SAS14_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
166	”SAS14_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
167	”SAS14_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
168	”SAS14_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
169	”SAS14_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
170	”SAS15_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
171	”SAS15_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
172	”SAS15_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
173	”SAS15_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
174	”SAS15_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
175	”SAS16_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
176	”SAS16_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
177	”SAS16_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
178	”SAS16_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
179	”SAS16_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0

Id	Name	Purpose	Type	Initial Value
180	”SAS17_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
181	”SAS17_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
182	”SAS17_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
183	”SAS17_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
184	”SAS17_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
185	”SAS18_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
186	”SAS18_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
187	”SAS18_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
188	”SAS18_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
189	”SAS18_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
190	”SAS19_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
191	”SAS19_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
192	”SAS19_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
193	”SAS19_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
194	”SAS19_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
195	”SAS20_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
196	”SAS20_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
197	”SAS20_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
198	”SAS20_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
199	”SAS20_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0

Id	Name	Purpose	Type	Initial Value
200	”AP_1_Name”	”Pathname of AP”	STRING_TYPE	””
201	”AP_1_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
202	”AP_1_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
203	”AP_1_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
204	”AP_1_ID”	”The AP identifier”	INTEGER_TYPE	0
205	”AP_2_Name”	”Pathname of AP”	STRING_TYPE	””
206	”AP_2_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
207	”AP_2_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
208	”AP_2_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
209	”AP_2_ID”	”The AP identifier”	INTEGER_TYPE	0
210	”AP_3_Name”	”Pathname of AP”	STRING_TYPE	””
211	”AP_3_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
212	”AP_3_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
213	”AP_3_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
214	”AP_3_ID”	”The AP identifier”	INTEGER_TYPE	0
215	”AP_4_Name”	”Pathname of AP”	STRING_TYPE	””
216	”AP_4_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
217	”AP_4_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
218	”AP_4_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
219	”AP_4_ID”	”The AP identifier”	INTEGER_TYPE	0
220	”AP_5_Name”	”Pathname of AP”	STRING_TYPE	””
221	”AP_5_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN

Id	Name	Purpose	Type	Initial Value
222	”AP_5_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
223	”AP_5_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
224	”AP_5_ID”	”The AP identifier”	INTEGER_TYPE	0
225	”AP_6_Name”	”Pathname of AP”	STRING_TYPE	””
226	”AP_6_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
227	”AP_6_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
228	”AP_6_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
229	”AP_6_ID”	”The AP identifier”	INTEGER_TYPE	0
230	”AP_7_Name”	”Pathname of AP”	STRING_TYPE	””
231	”AP_7_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
232	”AP_7_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
233	”AP_7_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
234	”AP_7_ID”	”The AP identifier”	INTEGER_TYPE	0
235	”AP_8_Name”	”Pathname of AP”	STRING_TYPE	””
236	”AP_8_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
237	”AP_8_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
238	”AP_8_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
239	”AP_8_ID”	”The AP identifier”	INTEGER_TYPE	0
240	”AP_9_Name”	”Pathname of AP”	STRING_TYPE	””
241	”AP_9_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
242	”AP_9_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
243	”AP_9_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””

Id	Name	Purpose	Type	Initial Value
244	”AP_9_ID”	”The AP identifier”	INTEGER_TYPE	0
245	”AP_10_Name”	”Pathname of AP”	STRING_TYPE	””
246	”AP_10_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
247	”AP_10_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
248	”AP_10_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
249	”AP_10_ID”	”The AP identifier”	INTEGER_TYPE	0
250	”AP_11_Name”	”Pathname of AP”	STRING_TYPE	””
251	”AP_11_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
252	”AP_11_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
253	”AP_11_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
254	”AP_11_ID”	”The AP identifier”	INTEGER_TYPE	0
255	”AP_12_Name”	”Pathname of AP”	STRING_TYPE	””
256	”AP_12_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
257	”AP_12_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
258	”AP_12_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
259	”AP_12_ID”	”The AP identifier”	INTEGER_TYPE	0
260	”AP_13_Name”	”Pathname of AP”	STRING_TYPE	””
261	”AP_13_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
262	”AP_13_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
263	”AP_13_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
264	”AP_13_ID”	”The AP identifier”	INTEGER_TYPE	0
265	”AP_14_Name”	”Pathname of AP”	STRING_TYPE	””

Id	Name	Purpose	Type	Initial Value
266	”AP_14_Status”	”Status (NOT_RUN, IN-ITIAL, RUNNING, SUS-PEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
267	”AP_14_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
268	”AP_14_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
269	”AP_14_ID”	”The AP identifier”	INTEGER_TYPE	0
270	”AP_15_Name”	”Pathname of AP”	STRING_TYPE	””
271	”AP_15_Status”	”Status (NOT_RUN, IN-ITIAL, RUNNING, SUS-PEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
272	”AP_15_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
273	”AP_15_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
274	”AP_15_ID”	”The AP identifier”	INTEGER_TYPE	0
275	”AP_16_Name”	”Pathname of AP”	STRING_TYPE	””
276	”AP_16_Status”	”Status (NOT_RUN, IN-ITIAL, RUNNING, SUS-PEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
277	”AP_16_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
278	”AP_16_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
279	”AP_16_ID”	”The AP identifier”	INTEGER_TYPE	0
280	”AP_17_Name”	”Pathname of AP”	STRING_TYPE	””
281	”AP_17_Status”	”Status (NOT_RUN, IN-ITIAL, RUNNING, SUS-PEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
282	”AP_17_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
283	”AP_17_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
284	”AP_17_ID”	”The AP identifier”	INTEGER_TYPE	0
285	”AP_18_Name”	”Pathname of AP”	STRING_TYPE	””
286	”AP_18_Status”	”Status (NOT_RUN, IN-ITIAL, RUNNING, SUS-PEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
287	”AP_18_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0

Id	Name	Purpose	Type	Initial Value
288	”AP_18_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
289	”AP_18_ID”	”The AP identifier”	INTEGER_TYPE	0
290	”AP_19_Name”	”Pathname of AP”	STRING_TYPE	””
291	”AP_19_Status”	”Status (NOT_RUN, IN-ITIAL, RUNNING, SUS-PEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
292	”AP_19_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
293	”AP_19_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
294	”AP_19_ID”	”The AP identifier”	INTEGER_TYPE	0
295	”AP_20_Name”	”Pathname of AP”	STRING_TYPE	””
296	”AP_20_Status”	”Status (NOT_RUN, IN-ITIAL, RUNNING, SUS-PEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
297	”AP_20_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
298	”AP_20_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
299	”AP_20_ID”	”The AP identifier”	INTEGER_TYPE	0

Id	Name	Purpose	Type	Initial Value
301	”Nb._Enditems”	”Number of enditems that can be monitored”	INTEGER_TYPE	0
302	”Nb._Enabled”	”Number of enditems enabled for monitoring”	INTEGER_TYPE	0
303	”Nb._Digital”	”No. of digital enditems that can be monitored”	INTEGER_TYPE	0
304	”Nb._Grp_Discr.”	”No. of group discrete enditems that can be monitored”	INTEGER_TYPE	0
305	”Nb._Analog”	”No. of analog enditems that can be monitored”	INTEGER_TYPE	0
306	”Nb._bytestream”	”No. of bytestream enditems that can be monitored”	INTEGER_TYPE	0
310	”Nb_Soft”	OOL Number of enditems currently out of soft limit(1)	INTEGER_TYPE	0
311	”Nb_Hard”	OOL Number of enditems currently out of hard limit (1)	INTEGER_TYPE	0
312	”Nb._Soft_exceptions ”	Number of soft limit violations since last START(1)	INTEGER_TYPE	0
313	”Nb._Hard_exceptions ”	Number of hard limit violations since last START(1)	INTEGER_TYPE	0
320	”Nb. Acquired”	Number of measurements currently acquired	INTEGER_TYPE	0
321	”Nb. EVL”	Number of acquired measurements or SW variables or derived values with EVL enabled	INTEGER_TYPE	0
322	”Nb. Measurements”	Number of meaurements defined/loaded to the test node	INTEGER_TYPE	0
323	”Nb. SW_Variables”	Number of software variables defined/loaded to the test node	INTEGER_TYPE	0
324	”Nb. Derived_Values	Number of derived values defined/loaded to the test node	INTEGER_TYPE	0

Id	Name	Purpose	Type	Initial Value
400	”Nb._stimuli”	”Number of GDUs sent out since last START”	INTEGER_TYPE	0
401	”Stimuli_Errors”	”Number of GDUs with errors in SAS/FEE since last START”	INTEGER_TYPE	0
402	”Wrong_Stimulus”	”Pathname of last stimulus that resulted in an error”	STRING_TYPE	””
404	”Destination”	”Destination of last erroneous simulus (SAS name)”	STRING_TYPE	””
410	”Nb._of_digital_GDUs”	”Number of digital output GDUs (loaded from MDB)”	INTEGER_TYPE	0
411	”Nb._of_analog_GDUs”	”Number of analog output GDUs (loaded from MDB)”	INTEGER_TYPE	0
412	”Nb._of_TC_GDUs”	”Number of GDUs with CCSDS TCs (loaded from MDB)”	INTEGER_TYPE	0
413	”Nb._of_GDUs”	”Total number of GDUs (loaded from MDB)”	INTEGER_TYPE	0
414	”Nb._of_bin.pack.GDUs”	”Number of binary packet GDUs (loaded from MDB)”	INTEGER_TYPE	0
415	”Nb.GDU_Verif_Succ”	Number of GDUs sent with successful verification	INTEGER_TYPE	0
416	”Nb.GDU_Verif_Fail”	Number of GDUs sent with failed verification	INTEGER_TYPE	0
420	”Nb._of_SW_cmd._sent”	”Number of SW commands sent”	INTEGER_TYPE	0
421	”Nb._of_rtn._packets”	”Number of return packets received for SW commands”	INTEGER_TYPE	0

REMARKS:

1. An enditem is only considered ”out–of–limits” if the limit is violated and the error count has been reached with respect to these housekeeping values. In the transition period (i.e. the limit is violated but the error count has not yet been reached) the housekeeping values will not change

Id	Name	Purpose	Type	Initial Value
500	”AP_21_Name”	”Pathname of AP”	STRING_TYPE	””
501	”AP_21_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
502	”AP_21_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
503	”AP_21_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
504	”AP_21_ID”	”The AP identifier”	INTEGER_TYPE	0
...	...	...	...	...
595	”AP_40_Name”	”Pathname of AP”	STRING_TYPE	””
596	”AP_40_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
597	”AP_40_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
598	”AP_40_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
599	”AP_40_ID”	”The AP identifier”	INTEGER_TYPE	0
601	”Served_WSs”	Number of connected workstations (HCI)	INTEGER_TYPE	0
700	”Nb.Cond_Enditems”	Number of Enditems carrying Conditions	INTEGER_TYPE	0
701	”Nb.Conditions”	Number of Conditions defined	INTEGER_TYPE	0
702	”Nb.TriggeredCond”	Number of actions triggered by conditions since last start	INTEGER_TYPE	0
900	”Time_of_Exception”	Time of last Monitoring exception (in seconds since midnight)	FLOAT_TYPE	0.0
901	”Time_of_Start_AP”	Time of last Start AP (in seconds since midnight)	FLOAT_TYPE	0.0
902	”Time_of_Resume_AP”	Time of last Resume AP (in seconds since midnight)	FLOAT_TYPE	0.0

Id	Name	Purpose	Type	Initial Value
1001	”PR1_Status”	”The status of printer1 (DISABLED, ENABLED, OTHERS)”	STATE_CODE_TYPE	$DISABLED
1002	”PR2_Status”	”The status of printer2 (DISABLED, ENABLED, OTHERS)”	STATE_CODE_TYPE	$DISABLED
1003	”PQ1_Status”	”Status (OFF, READY, PRINTING, NO_PAPER, NO_TONER)”	STATE_CODE_TYPE	$OFF
1004	”PQ2_Status”	”Status (OFF, READY, PRINTING, NO_PAPER, NO_TONER)”	STATE_CODE_TYPE	$OFF
1005	”PQ1_Jobs”	”The number of jobs in print queue 1”	INTEGER_TYPE	0
1006	”PQ2_Jobs”	”The number of jobs in print queue 2”	INTEGER_TYPE	0
1011	”MD_Free_Spac”	”The free disc space on the magnetic disc in kilobytes”	INTEGER_TYPE	0
1012	”FA_1_Dev_Status”	”The device status (OK, NOT_OK) of the long term storage medium”	STATE_CODE_TYPE	$OK
1020	”TRDB_Over-all_status”	”The overal status of the test result data base (OK, NOT_OK)”	STATE_CODE_TYPE	$OK
1050	”Session_Name”	”The name of the current test execution session”	STRING_TYPE	””
1051	”TRDB_Eval_U”	”The number of evaluation users connected to TRDB”	INTEGER_TYPE	0
1060	”TRDB_Event_Usage”	”The percentage of event table space used”	INTEGER_TYPE	0
1061	”TRDB_MA_Usage”	”The percentage of master archive table space used”	INTEGER_TYPE	0
1062	”TRDB_Misc_Usage”	”The percentage of miscellaneous table space used”	INTEGER_TYPE	0
1100	”T_SYNC”	”The local clock is synchronized with NTP (FALSE, TRUE)”	STATE_CODE_TYPE	$FALSE
1101	”MTU_PRESENT”	”Has an external master time unit (FALSE, TRUE)”	STATE_CODE_TYPE	$FALSE

Id	Name	Purpose	Type	Initial Value
1102	”SMT_STATUS”	”The status of the SMT (NOT_INIT, STOPPED, RUNNING)”	STATE_CODE_TYPE	$NOT_INIT
1103	”SYSTEM_LT_SYNC_STATUS”	”The system synchronisation status of the LT (only on Master valid)” <u>Alternative</u> <u>meaning</u> UNDEFINE no valid statement possible TRUE system is synchronized FALSE system is not synchronized ConnLost one (or more) client don’t answer NoUpdate the local information will not be updated NoMaster the current node is not time domain master	STATE_CODE_TYPE	$UNDEFINE

7.2 Preparing Special Application Software

7.2.1 Introduction

7.2.1.1 SAS Concept in CGS

This chapter covers two topics:

- how to implement an SAS and
- how to command an TES SAS in the CGS environment

Main focus will be put on guide-lines how special application software (SAS) shall be written in the CGS context, the second point will be mentioned briefly.

For CGS, SAS can be developed for different purposes:

- TES SAS
  - to interface between frontends and the TES process running on test nodes
  - to process data on top of the standard CGS facilities during ongoing tests
- TEV SAS
  - to evaluate data on top of the standard CGS evalution features
- DBS SAS
  - to implement specific final archiving tools complementing the standard FA_SAS which allows to export/import data to/from magneto–optical disks

7.2.1.1.1 TES SAS

TES SAS constitutes the data interface between CGS and the front end equipment (FEE) directly interfacing to the test article, referred to as ’unit under test’ (UUT) here. Also, TES SAS adds special data processing features to a CGS system which the user needs in order to run the tests.

The general concept in CGS is that the interface to the unit under test (UUT) is established by special front end equipment (FEE) providing the direct interface on hardware level plus special application software (SAS) providing the ’glue’ between the FFE and CGS itself. Through this interface, the data from the UUT are acquired in raw data format and stimuli are sent to it.

Two different acquisition processes are supported by the SAS:

- data acquisition from the UUT
  i.e. the SAS acquires user selected data from the UUT and commands the UUT via the FEE at the speed of the employed hardware
- data delivery to CGS
  i.e. SAS delivers the acquired UUT data packetized in ADUs to CGS and receives commands contained in GDUs from CGS, both according to user specifications laid down in the related ADU and GDU descriptions

In CGS, the product TES provides this interface to SAS in the form of an application program interface (API). The TES_API is in fact a collection of Ada packages which have to be included into the SAS code

during compilation. Thus, a high level interface on procedural level is constituted facilitating easy use and reliable data exchange.

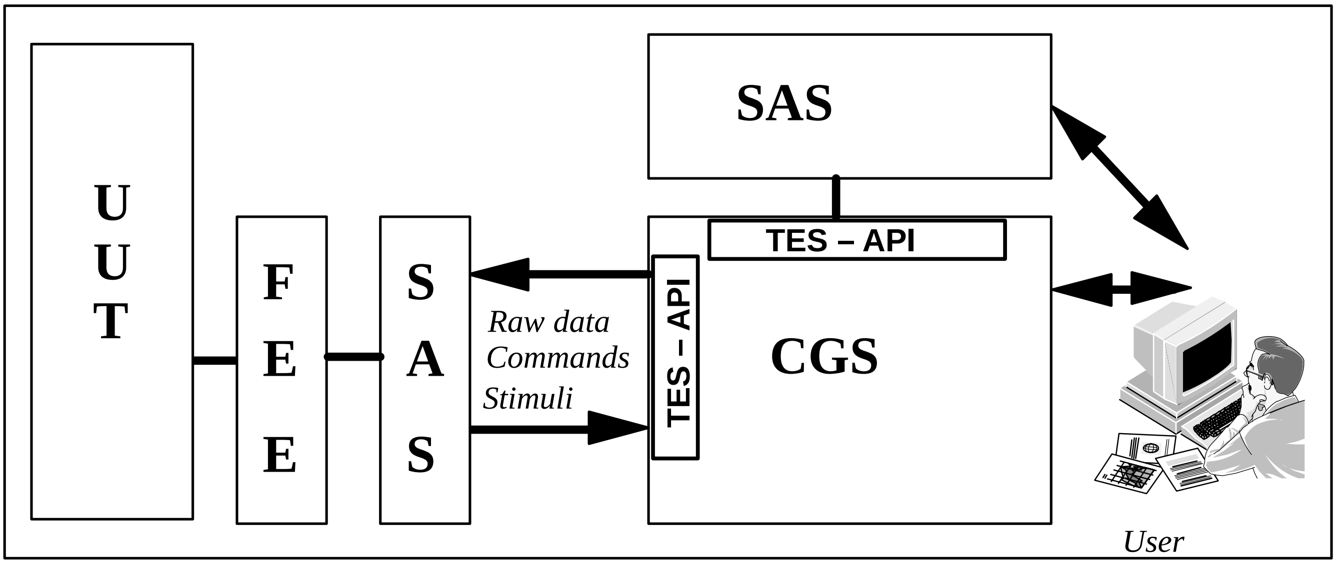


Figure 7–23 : *The overall concept of TES SAS together with CGS*

In addition, SAS is foreseen in all those areas where the user needs very special data processing features not provided by CGS. This can be special data visualisation, storing or calculations.

SAS are separate operating system processes running under CGS control and communicating with CGS via standardised, internal mechanisms. The SAS programmer does not have to care for the low level communication with CGS because this is hidden inside the API.

7.2.1.1.2 TEV SAS

TEV SAS constitutes the Softare on top of CGS evaluation facilities. It adds special data processing features to a CGS system which the user needs in order to evaluate the data stored during tests.

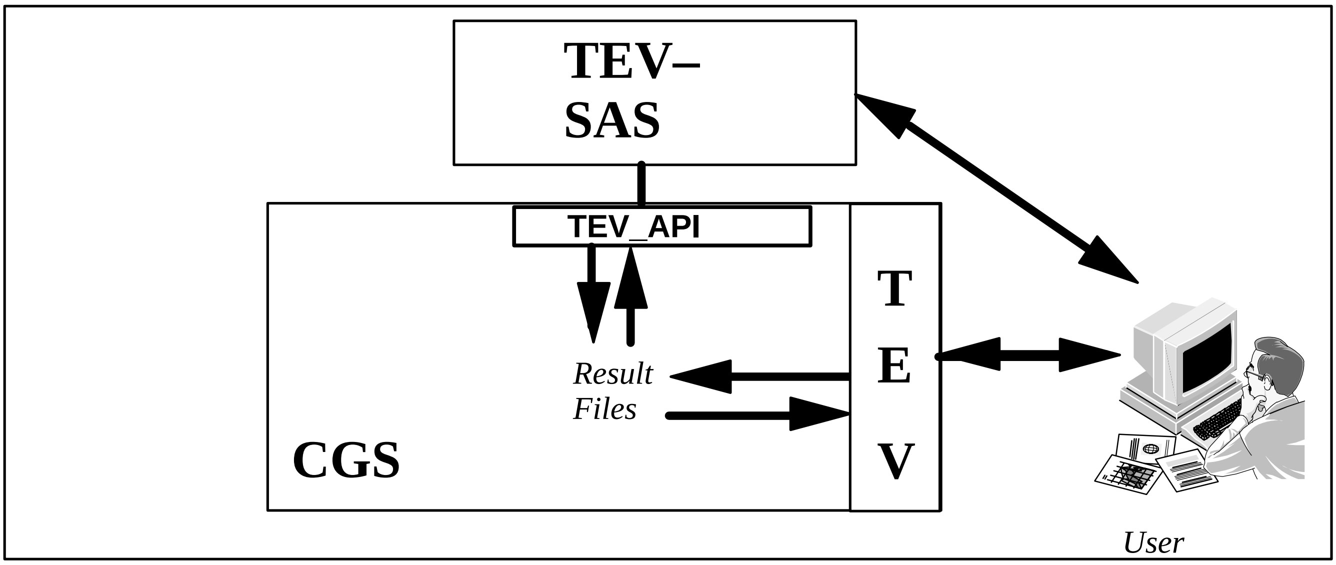


Figure 7–24 : *The overall concept of TEV SAS together with CGS*

7.2.1.1.3 DBS SAS

DBS SAS constitutes the Softare on top of CGS to implement data storage to additional media/devices. It allows to implement additional SAS as driver for specific final archive devices.

7.2.1.2 SAS Implementation Rules

Special Application Software (SAS) are programs written in Ada which have to run on HP test nodes. For the development of an SAS the Ada Compiler System Environment from the company Alsys is used.

Part of the CGS delivery is the directory $GSAF_HOME/cgs/lib/cgs_api which contains subdirectories for each architecture supported (sun5/hp_ux8). Each of these contains the following

- the subdirectory EXT_ICD.<architecture>.source_tree containing the Ada and C source files of the API
- the file EXT_ICD.<architecture>, a list of the Ada source files describing the API (Ada Specifications)
- the file EXT_ICD.<architecture>.closure, a list of the Ada source files implementing the whole API
- the file makefile.<architecture> compiling the Ada source files and
- a README file explaining how to use CGS_API
- the bin directory containing link scripts for each type of SAS. These script are to be called for a sas giving the name of the SAS main program as first parameter (e.g. link_tes_sas_sun SAS_1)
    - link_dbs_sas
      script to link a DBS SAS on Sun
    - link_tes_sas_hp
      script to link a TES SAS on HP
    - link_tes_sas_sun
      script to link a TES SAS on Sun
    - link_tev_sas
      script to link a TEV SAS on Sun
    - linkada
      general script to be called by link_tev_sas (links any program by inlcuding the Oracle Interface Libraries)

SAS programs that use the CGS_API should be developed in a separate Ada library. The CGS_API Ada library must be found in the parent library tree.

NO TAG illustrates the set-up of Ada libraries necessary to implement SAS. On the root level there has to be the standard library provided by Alsys. The TES_API (Test Execution Software Application Programming Interface) provides the interface to CGS for SASs. Therefore this library has to be the parent library for all SASs. If several SAS are developed in parallel and contain common parts it is recommended to create an additional SAS common library.

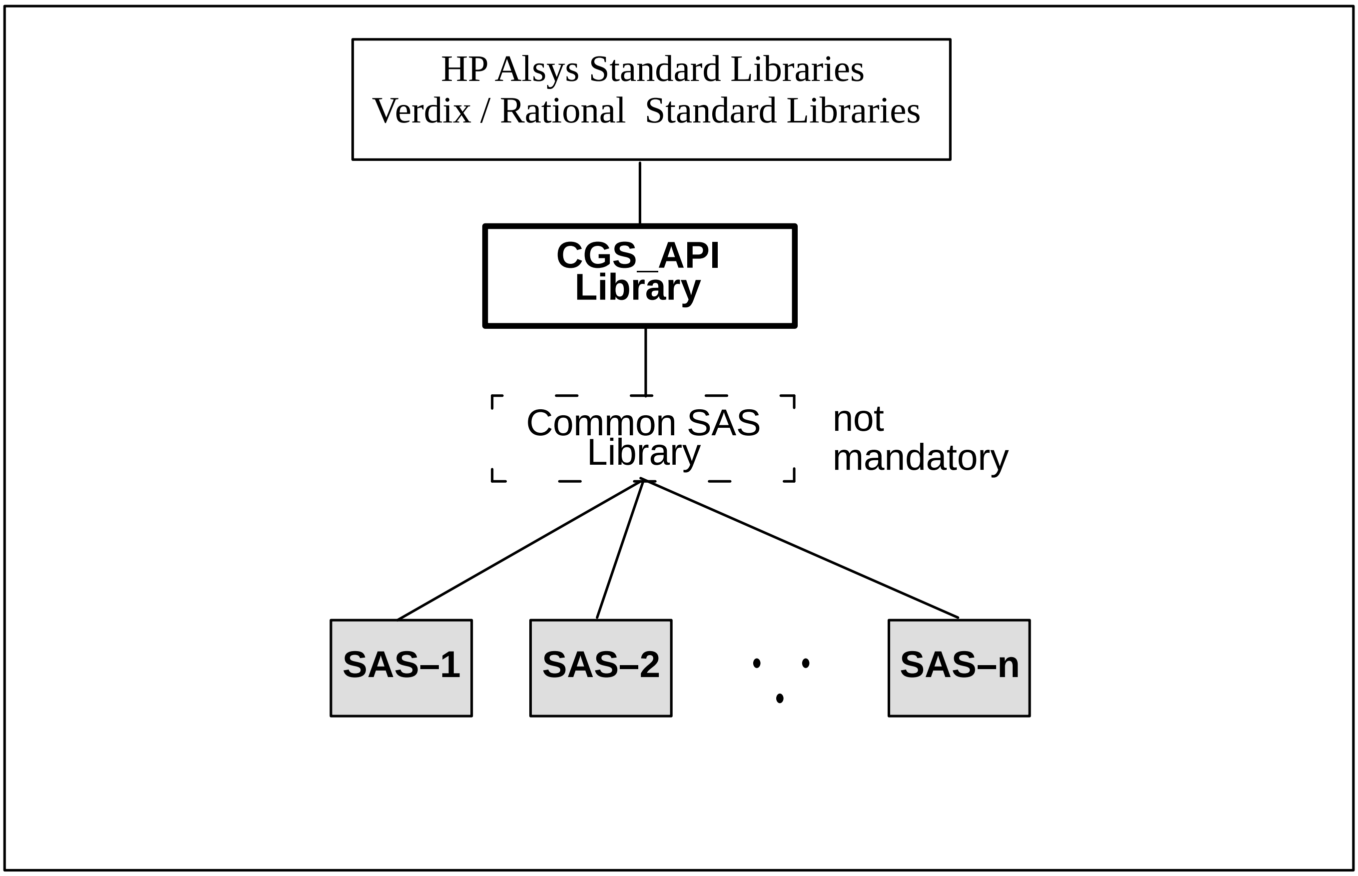


Figure 7–25 : *Ada Library Structure for SAS Development – this picture shows the visibility rule, not a real directory structure*

As already mentioned before, SAS constitute the real–time data and command interface to CGS via functions / procedures of packages in the TES_ API library or perform the specific calculations needed during on–line testing. Concerning the operational behaviour of the SAS, the main difference is whether or not it will internally contain ADA tasking or not. As such, the two different types of SAS are:

- Synchronous SAS, which wait for any CGS command in a permanent main loop inside the main program, then process it and after having processed it successfully, again start waiting for another command. These SAS could for example display raw data packets routed from other test nodes to workstations in a very special way or they could do special calibrations or limit checking of end item data.

  Synchronous SAS do not need Ada tasking inside the user written code (and we assume throughout this document that they will not contain Ada tasking!) since they sequentially perform a series of actions under CGS control.

- Asynchronous SAS, which have to handle requests from CGS as well as the front end equipment(s) in an asynchronous way do not know when and which event to handle. Usually, these SAS would include Ada tasking to handle the different events asynchronously. Possibly, UNIX interrupts are bound to task entries or special event handling routines (interrupt handler) are used.

Asynchronous SAS are restricted to the limitations imposed by the Ada run-time environment provided with Alsys compiler. These restriction are mainly how asynchronous events have to be handled by the Ada run-time system and how Ada task scheduling is done.

In the following the main program structure of the two categories of SAS programs, synchronous and asynchronous will be explained in more detail. In order to understand the overall program logic it is important to note here that the TES API provides a procedure READ_COMMAND which returns a command from CGS to the SAS. Through this interface the SAS is controlled and all commands and data requests are routed to the SAS.

7.2.2 The SAS Main Program

The overall structure of a SAS 'main program' depends on the environment the SAS is intended for and also on the desired operational behavior.

In general, two different environments are envisaged:

- a non X–View environment for all SAS running on the HP node. Normally, these SASes constitute the real–time data and command interface to CGS via the product TES or perform the specific calculations needed during on–line testing. Additionally, those SASes running on a SUN platform without having an X–Views (or OPENLOOK) user interface also fall into this category.
- an X–View environment for those SASes running on the SUN, assuming these SASes have an own user interface using the X–VIEW and/or OPENLOOK style. In this case, the program has to include the so–called 'window main loop', a general purpose event handler provided by X/OPENLOOK. The main purpose of these SASes, as envisaged currently is the special visualisation of test data, special evaluations, etc.

Concerning the operational behavior of the SAS, the main difference is whether or not it will internally contain ADA tasking or not. As such, the two different types of SAS are:

- synchronous SASes, which in a permanent main loop inside the main program wait for any CGS command to come, then process it and after having processed it successfully, again start waiting for another command. These SASes could for example display raw data packets routed from other test nodes to workstations in a very special way or they could do special calibrations or limit checking of enditem data.
  These SASes do not need Ada tasking inside the user written code (and we assume throughout this document that they will not contain Ada tasking!) since they sequentially perform a series of actions under CGS control.
- asynchronous SASes, which have to handle requests from CGS as well as the front end equipment(s) in an asynchronous way not know when and which event to handle. Usually, these SASes would include Ada tasking to handle the different events asynchronously. Possibly, UNIX interrupts are bound to task entries or special event handling routines (interrupt handler) are used.

Not all possible combinations of the afore mentioned criteria will make sense due to the limitations imposed by the X–VIEW environment as well as the Ada runtime environment provided with a given compiler. These restriction are mainly how asynchronous events have to be handled by the Ada runtime system and how Ada task scheduling is done.NO TAG lists the allowed combinations and restrictions:

	non X–VIEW	X–VIEW
synchronous	allowed, no restrictions	allowed, XVIEW restrictions
asynchronous	allowed, Ada run–time restrictions	allowed, XVIEW restrictions

Table 7–1 : *Restrictions in the ADA X–VIEW Environment*

7.2.2.1 General Structure in a non–X–View Environment

In a non X–View environment, the Ada main program structure is only limited by the scheduling capabilities provided by the chosen compiler, i.e. the Ada run–time system, with respect to handling asynchronous IO.

In this chapter the main program structure of the two main categories of SAS programs, synchronous and asynchronous will be explained.

In order to understand the overall program logic it is important to note here that the TES API provides a procedure READ_COMMAND which returns a command from CGS to the SAS. Through this interface the SAS is controlled and all commands and data requests are routed to the SAS.

7.2.2.1.1 Synchronous SAS Main Program Structure

Synchronous SAS are those explicitly waiting for commands from CGS and then reacting to them without doing other processing in parallel (e.g. device IO). As such, the main program structure of this type of SAS could be as follows:

Synchronous SAS Main Program Structure

```
with TES_API;                                  — this package provides the TES_API operations !
with ADT_TES_TO_SAS_COMMAND;                   — this package provides a high level interface for
...                                            —  handling the commands sent from TES to SAS
procedure MAIN is                              — here the main program starts
CMD: ADT_TES_TO_SAS_COMMAND.T_COMMAND;         — the variable for the command

begin
    ...                                        — do some initializations
    loop                                       — the main program loop starts here
       ...                                     — do some more initializations
       TES_API.READ_COMMAND(...,               — wait for a command and block
                        COMMAND => CMD,
                        ...,
                        BLOCK => TRUE);
       process_command(...);                   — and process the command, e.g. the device IO
                                               — occurs completely !
    end loop                                   — the main program loop ends here
end MAIN;                                      — here the main program ends
```

This program really waits for a command from the CGS Test Execution Facility, e.g. a request to send a stimulus (GDU request) and then processes it, resulting in device IO. Only if all processing is finished, then the ’main loop’ of the program starts from the beginning again, waiting for the next command.

The above kind of program logic does not allow for any periodical processing to take place inside the SAS, e.g. a cyclical IO with a device to be controlled. If this is desired, a pseudo asynchronous behaviour can be achieved using the following main program structure:

Pseudo asynchronous SAS Main Program Structure

```
with TES_API;                                   — this package provides the TES_API operations !
with ADT_TES_TO_SAS_COMMAND;                    — this package provides a high level interface for
...                                             —  handling the commands sent from TES to SAS
procedure MAIN is
    — here the main program starts
CMD: ADT_TES_TO_SAS_COMMAND.T_COMMAND;          — the variable for the command
DLY: duration;                                  — the overall cycle time for this SAS
```

to be continued...

```
begin
    ...                                         — do some initializations
    loop                                        — the main program loop starts here
      delay(DLY)                                — delay for the overall cycle time
      perform_cyclic_operations(...);           — do the regular job here
      TES_API.READ_COMMAND(...,                 — check for a command, do not block
                        COMMAND => CMD,
                        ...,
                        BLOCK => FALSE);
      if (ADT_TES_TO_SAS_COMMAND.COMMAND_ALTERNATIVE(CMD) <>
        ADT_TES_TO_SAS_COMMAND.NO_COMMAND) then
        process_command(...);                   — process command if any
      end if;
    end loop                                    — the main program loop ends here
end MAIN;                                       — here the main program ends
```

7.2.2.1.2 Asynchronous SAS Main Program Structure

If the handling of data inside the SAS has to be really asynchronous to the handling of the communication to TES, then another program logic has to be chosen, using the Ada tasking mechanisms.

A possible logic for this kind of application is the following:

Asynchronous TES SAS Main Program Structure

```
with TES_API;                                 — this package provides the TES_API operations !
with ADT_TES_TO_SAS_COMMAND;                  — this package provides a high level interface for
...                                           —  handling the commands sent from TES to SAS
procedure MAIN is                             — here the main program starts
CMD: ADT_TES_TO_SAS_COMMAND.T_COMMAND;            — the variable for the command
DLY: duration;                                — the overall cycle time for this SAS
task CMD_HANDLER is                           — specify a task to handle the commands
entry START;                                  — accept an entry to start the task
      ...                                     — possibly some more entries
end CMD_HANDLER;                              — end of this task specification

task PROCESSOR is                             — specify a task to process data asynchronously
entry START;                                  — accept an entry to start the task
entry DO_SOMETHING;                           — an entry to do some processing, e.g. a GDU
      ...                                     — possibly some more entries
end PROCESSOR;                                — end of this task specification
```

– –

to be continued...

```
task body CMD_HANDLER is                          — body of the CMD_HANDLER task
begin
      accept START;                               — accept an entry to start the task
      ...                                         — do some initializations
      loop                                        — the main loop starts here
         TES_API.READ_COMMAND(...,                — check for a command, block
                           COMMAND => CMD,
                           ...,
                           BLOCK => TRUE);
         process_command(...);                    — process command, possibly rendezvous with
                                                  — PROCESSOR.DO_SOMETHING;
      end loop                                    — the main loop ends here
end CMD_HANDLER;                                  — end of the body for CMD_HANDLER

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

task body PROCESSOR is                               — body of the PROCESSOR task
begin
      accept START;
         — accept an entry to start the task
      ...                                         — do some initializations
      loop                                        — the main loop starts here
         select                                      — check which rendezvous is desired
            accept DO_SOMETHING do ...end accept;    — in case of DO_SOMETHING do it
            or
            accept ...                            — another rendezvous
         else
            do_something_else                     — no rendezvous, do something special,
                                                  — e.g. provide ADUs cyclically
         end select;
      end loop;                                   — the main loop ends here
end PROCESSOR;                                    — end of the body for PROCESSOR

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

begin                                             — here the main program starts
      CMD_HANDLER.START;                          — activate the CMD_HANDLER
      PROCESSOR.START;
— activate the PROCESSOR
end MAIN;                                         — here the main program ends
```

7.2.2.2 General Structure in an X–View Environment

In an X–VIEW environment, the ’main program’ has to pass control at some point in time to the window main loop which then acts as an event dispatcher taking control of the program logic. Because of this the distinction between synchronous and asynchronous SASes does not make much sense from an implementation point of view, since it is always the window main loop passing the control to the relevant piece of software inside the main program.

To perform the various actions in an X–VIEW based program, so–called ’call back procedures’ have to be defined which perform the relevant tasks, e.g. perform a certain IO with a certain device and store the data somewhere. These call back procedures have to be linked to certain active screen elements so that in case the user touches these, the call back procedure is ’called’ by the window main loop.

Care has to be taken that the processing done in the call back procedure does not block the window main loop to avoid deadlock situations.

Because the XView toolkit and the standard Sun Ada runtime system are using the same Unix signals in conflict a special runtime system together with the XView Ada Interface is provided.

To solve the signal conflicts the Ada kernel sets up notify timer and signal handlers at initialization time, i.e. signal handling (for doing delays, servicing timeouts and task time slicing) is done in the scope of the XView notifier. The Notifier dispatches the signals either to the Ada kernel or to the Ada program (e.g. to call back; see Figure 23).

Therefore, if an event handler (or callback) blocks, it also blocks all Notifier activity.

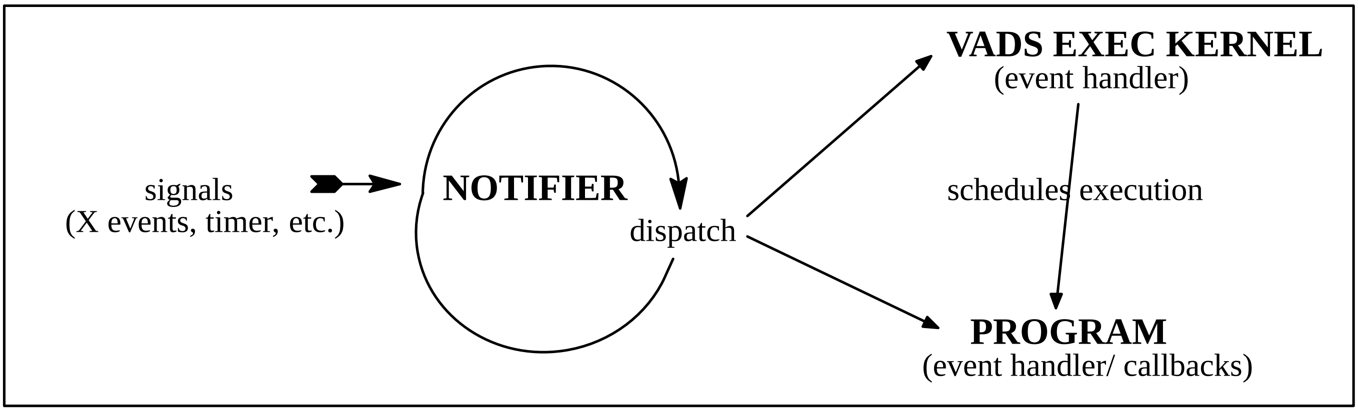


Figure 7–26 : *Event Dispatch*

To synchronize access to the notifier all calls to the notify (and window) services must be protected via the IN_NOTIFY semaphore. The Notifier protects itself by enter/leave IN_NOTIFY preceded/followed by a call to dispatch, i.e. event handlers and callbacks are already IN_NOTIFY.

NOTE: Since enters/leaves of IN_NOTIFY semaphore can be nested from the *same* task, Notifier event handlers/callbacks may call enter/leave.

In some situations, however, it is possible to block the notifier.

The following bullets point out situations on which the notifier blocks, when used within event handlers or callbacks:

- **delay** A_DURATION;
  The notifier is suspended until the notifier has called the timer routine for doing delays (dead lock).

- TASK_NAME.ENTRY_NAME; — call rendezvous
  This statement will suspend the notifier if TASK_NAME is not ready to accept ENTRY_NAME. The system blocks because the notifier has to call the task time slicing service to continue TASK_NAME until it's ready to accept the rendezvous.

The system will also block if a task has entered IN_NOTIFY and notifier activity is required due to delay, time slicing, etc. , because the Notifier can't call *dispatch* until the task has released the semaphore.

In order to avoid blocking, Ada task rendezvous have to be implemented in a special way.

Ada provides a select statement implementing a conditional entry call (see Ada LRM, section 9.7.2) that issues an entry call that is then canceled if a rendezvous is not immediately possible. This statement can be used within event handlers/callbacks without blocking the system, e.g.

```
task body TASK_NAME is
begin
   loop
      accept LOAD;
      DO_LOAD;
   end loop;
end TASK_NAME;
procedure CALLBACK is
begin
   select
      TASK_NAME.LOAD;
   else
      null; -- ignore the load command
   end select;
exception
   when others =>
          -- consume all exceptions, because propagated
          -- exceptions will terminate the notifier
     null;
end CALLBACK;
```

In order to facilitate save access to X–VIEW services and to avoid blocking situations, a package X_SYNC, calling the N_NOTIFY enter/leave operations internally, could be defined as follows:

```
package X_SYNC is
   procedure ENTER;
   -- DESCRIPTION
          -- Enters a semaphore which protects the notifier's critical
          -- region and the access to the X server. This operation must
          -- precede any call to notifier/window services including
          -- routines that may access the X server like DataViews
          -- operations. Refer to XView Interface and Runtime System,
          -- section 4.5 for more details.
   procedure LEAVE;
   -- DESCRIPTION
          -- Leaves a semaphore which protects the notifier's critical
          -- region. This operation must follow any call to
          -- notifier/window services. Refer to XView Interface and
          -- Runtime System, section 4.5 for more details.
end X_SYNC;
```

The X_SYNC package calls directly the window/notifier operations enter/leave. For the operations the pragma INLINE is set to reduce the amount of procedure calls.

7.2.2.3 Handling Synchronous IO with Front End Devices

Synchronous IO with front end devices in this context means that the SAS takes the initiative and starts 'talking' to the device and then waits for it to respond. During this process, the SAS is blocked for other activities, i.e. communication with CGS.

This type of IO handling always works without problems in all environment, be it X–Views or UNIX solely. However, in the X–VIEW environment the general constraints of the window main loop have to be considered, as described already in section NO TAG.

The general disadvantage of being blocked during the device IO, especially during the read from the external device, can be overcome by introducing asynchronous device IO, as it will be described in the next chapter.

7.2.2.4 Handling Asynchronous IO with Front End Devices

In asynchronous device IO mode, the SAS does not explicitly wait for device IO to occur but instead sets up a processing scheme internally in a way that IO can be handled whenever it occurs.

Usually, this is implemented using interrupt service routines, X–VIEW notifier procedures or Ada tasking. Which method can be used depends of course on the type of application the SAS is, i.e. whether it is an X–VIEW program or not, but also on the chosen compiler/runtime system. Many compilers do not correctly implement the asynchronous IO in the tasking environment.

Assuming the compiler vendor has delivered a runtime system being able of correctly handling asynchronous device IO, then the following general architectures of a SAS program exist:

- the SAS includes one or more tasks which are bound to device IO interrupts with some of their entries
- the SAS includes one or more tasks which are blocked in READs on the respective input lines
- the SAS includes X–VIEW notifier procedures which are invoked by the runtime system on the occurrence of an IO.

7.2.3 How to implement SAS

Before starting the implementation, a proper environment set-up has to be ensured in terms of setting-up

- the UNIX environment
- the compiler system environment
- the system topology table (for TES_SAS)

For the UNIX environment some variables have to be set and paths must be visible in the users search path. Some of the required conditions of the Ada compiler system environment has already been mentioned above, but the detailed data and additional information are given in the following procedure.

The system topology table is dependant on the environment at users site, therefore the set-up of the table is described in the installation manual.

Note that the SYSTEM_TOPOLOGY_TABLE is maintained by the **CGS administrator** only (See section on test setup using the TSCV tool).

7.2.3.1 Implementing a TES_SAS

The following procedure describes the necessary steps to setup the compilation environmnent and to compile/link a TES SAS.

Preparing the SAS environment and Implementing TES SAS on HP

- **Check** with your system administrator or tool responsible that the **Alsys Compiler System V 5.5.4** is installed on your HP machine correctly.
- **Login as a cgs user and open** a C-shell-tool. Verify your ALSYS ada aenvironment (LM_LICENSE_FILE, ADA_PATH, ALSYCOMP_DIR,POSIX_LIBRARY)
- **set path = ($path $ALSYCOMP_DIR)**
- **set API_LIB =** $GSAF_HOME/cgs/lib/cgs_api
- **Go** to the directory $API_LIB and **read** the READ_ME file for more information.
- **Create your own Ada Family and your own Ada Library**
- ada.mkfam TEST_TES_API
- ada.lmgrTEST_TES_API
- Library_Manager.copy
- FROM => $API_LIB/hp_ux8/lib.export
- TO => TES_API_LIB
- MODE=> IMPORT
- Library_Manager.new PARENT => TES_API_LIB
- LIBRARY => <your library name>
- Library_Manager.quit
- **Compile your own SAS sources**
- Copy your sources to the directory <your library name>
- adacli
- Ada.compile
- SOURCE => $API_LIB/<your library name>/<source_file>
- LIBRARY => <your library name>
- Ada.quit
- **Link your own SAS objetcs**
- *The script link_tes_sas_on_hp might need adaptation to define the right name for the executable, or alternatively, copy the script to your own ”link_your_sas” script, modify and use this one.*
- $API_LIB/hp_ux8/bin/link_tes_sas_hp   <your_sas_main_program>
- **Copy the executable to the CGS SAS directory**
- su   cgsadmin
- cp   <your_sas> $GSAF_HOME/sas/bin
- If the SAS has data files accompanied, **ensure** that all these are placed in or linked to the $GSAF_HOME/sas/data directory.

Preparing the SAS environment and Implementing TES SAS on SUN

- **Check** with your system administrator or tool responsible that the Sun Ada/Verdix Compiler 6.2.3c is installed on your Sun machine correctly. Check that the CGS_API has been compiled on Sun acc. to the instructions given in the CGS installation manual or CGS release notes.
- **Login as a cgs user and open** a C-shell-tool. Verify your ada environment
- **>set path = ($path <Ada Compiler Home>/vads/bin)**
- note: The compiler path can be obtained by:
- >more /etc/VADS
  sample output: SELF_TARGET:6.2:/vobsun2_tools1/vads/self: # Sun Solaris 2.x–host
  where <compiler–path> is the 3rd parameter of the output above without "/self"
- >setenv ADA_HOME <compiler–path>
  > cd $GSAF_HOME/cgs/lib/cgs_api/sun5
- **setenv API_LIB =** $GSAF_HOME/cgs/lib/cgs_api
- **Go** to the directory $API_LIB and **read** the READ_ME file for more information.
- **Create your own Ada Library**
- a.mklib TES_API_LIB $API_LIB/sun_adalib
- **Compile your own SAS sources**
- Copy your sources to the directory $API_LIB/TES_API_LIB
- a.make –v –f *.a
- **Link your own SAS objetcs**
- *The script link_tes_sas_on_sun might need adaptation to define the right name for the executable, or alternatively, copy the script to your own "link_your_sas" script, modify and use this one.*
- $API_LIB/sun5/bin/link_tes_sas_sun <your_sas_main_program>
- **Copy the executable** <your_sas_main_program> **to the CGS SAS directory**
- su cgsadmin
- cp <your_sas> $GSAF_HOME/sas/bin
- If the SAS has data files accompanied, **ensure** that all these are placed in or linked to the $GSAF_HOME/sas/data directory.

Define the SAS inside the MDB

- **Create** an **end item** of type EGSE_SOFTWARE according section 6.2., and enter the required information.
- If the name of the SAS does not match the name of the corresponding end item in your MDB instance, **rename** the SAS executable file.
- Contact your system administrator to enter your SAS name and the port number in the SYSTEM_TOPOLOGY_TABLE.

The above described procedure s represent the general procedure for TES SAS development and how to cope with the resulting executable program. Also the structure of the different kind of SAS main programs

have been described in the section before. So what is still missing, is the description of the provided CGS interface itself, i.e. the description of functions and procedure provided by the CGS TES API.

The Application Programming interface of the CGS Test Execution Facility is described in detail in appendix F of this document.

7.2.3.2 Implementing a TEV_SAS

Preparing the SAS environment and Implementing TEV SAS (on SUN)

- **Check** with your system administrator or tool responsible that the Sun Ada/Verdix Compiler 6.2.3c is installed on your Sun machine correctly.
- **Login as a cgs user and open** a C-shell-tool. Verify your Verdix ada aenvironment
- **set path = ($path <Verdix_home>/vads/bin)**
- **set API_LIB =** $GSAF_HOME/cgs/lib/cgs_api
- **Go** to the directory $API_LIB and **read** the READ_ME file for more information.
- **Create your own Ada Library**
- a.mklib TEV_API_LIB $API_LIB/sun_adalib
- **Compile your own SAS sources**
- Copy your sources to the directory $API_LIB/TEV_API_LIB
- a.make –v –f *.a
- **Link your own SAS objetcs**
- *The script link_tev_sas might need adaptation to define the right name for the executable, or alternatively, copy the script to your own ”link_your_sas” script, modify and use this one.*
- $API_LIB/sun5/bin/link_tev_sas <your_sas_main_program>
- To startup the TEV SAS, just enter the name of the executable within the C–shell Tool.

7.2.3.3 Implementing a DBS_SAS

Preparing the SAS environment and Implementing DBS SAS (on SUN)

- **Check** with your system administrator or tool responsible that the Sun Ada/Verdix Compiler 6.2.3c is installed on your Sun machine correctly.
- **Login as a cgs user and open** a C-shell-tool. Verify your Verdix ada aenvironment
- **set path = ($path <Verdix_home>/vads/bin)**
- **set API_LIB =** $GSAF_HOME/cgs/lib/cgs_api
- **Go** to the directory $API_LIB and **read** the READ_ME file for more information.
- **Create your own Ada Library**
- a.mklib DBS_API_LIB $API_LIB/sun_adalib
- **Compile your own SAS sources**
- Copy your sources to the directory $API_LIB/DBS_API_LIB
- a.make –v –f *.a
- **Link your own SAS objetcs**
- *The script link_tev_sas might need adaptation to define the right name for the executable, or alternatively, copy the script to your own ”link_your_sas” script, modify and use this one.*
- $API_LIB/sun5/bin/link_dbs_sas <your_sas_main_program>
- To startup the DBS SAS, just enter the name of the executable within the C–shell Tool.
- To implement the DBS SAS as a new driver for a final archive device in CGS, edit the script $DBS_HOME/bin/common/start_fa_sas to startup the new SAS instead of the $DBS_HOME/bin/sun5/dbs_fa_sas program.

7.2.4 How to control TES SAS

TES SAS can be controlled either by APs containing the command sequence or interactive using the TES HLCL command line window.

A general description is given using the SAS state transition diagram.

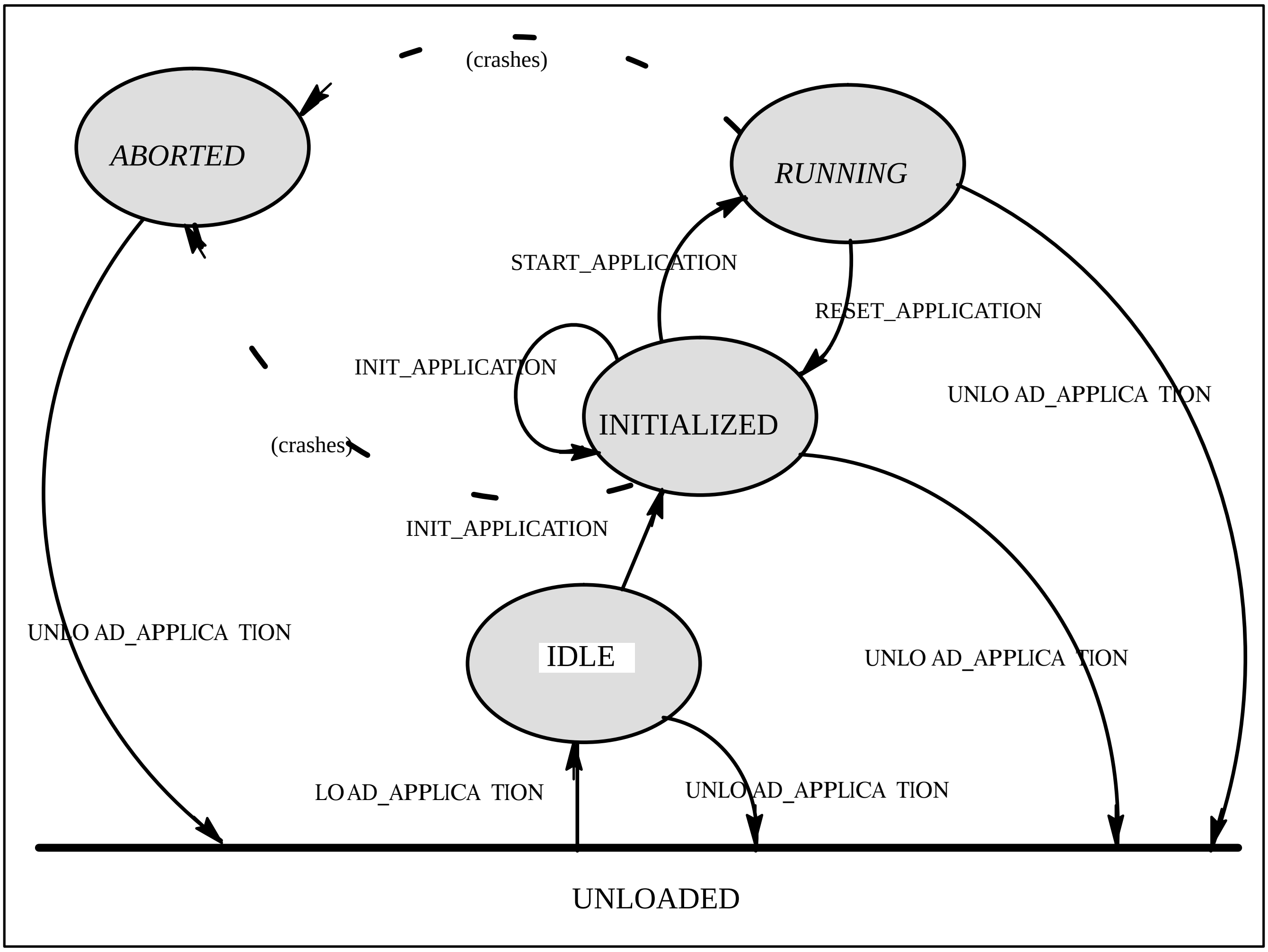


Figure 7–27 *: Relationship of SAS state transitions*

TES SAS can be loaded (i.e. created as a UNIX process) via the UCL library command

- LOAD_APPLICATION

This command loads the SAS to the local test node or on a workstation. Also a Simulation Node or the DB Server node may be selected.

It is possible to transfer parameter to the SAS via the LOAD_APPLICATION command. These parameter are transfered on the command line and can be read by the SAS via the standard mechanisms (i.e. as if the parameter were given via a UNIX shell command).

Starting a SAS on a workstation may need to transfer within the parameter block the TES identification, where the SAS shall connect to: e.g.: ”<SAS Param1> TES_xx”

The following constraints need to be followed:

- the TES identification must always be the identification of the test node where the SAS was started
- the SAS must not be started on another test node, where an entry exists in the System Topology Table for a TES_xx, independently, if a TES is running on this test node or not. (The TES_API will be confused, if the SAS is started on such a test node)

TES SAS can enter different states. After loading the SAS enters the state IDLE. The operator can now control the state of the SAS by giving the command

- INIT_APPLICATION

The SAS will initialize itself according to the parameter given and then enter the state INITIALIZED, if no serious error occurs (the SAS ”crashes”) , which prevents the SAS from initializing correctly. In this case the SAS enters the state ABORTED.
Normally the SAS will now remain in the state INITIALIZED until the

- START_APPLICATION

command has been received. The SAS will then enter the RUNNING state and perform its foreseen task. (If an error occurs the SAS goes to the state ABORTED.)

AS soon as the SAS is running commands requesting data can be send as well as commands to the UUT. (These commands with a description how to use them can be found in the UCL system library).
If the SAS is delivering ADUs, it will announce the ADU service after entering the RUNNING state. If the SAS is accepting GDUs, it will accept these in the RUNNING state only, after having announced the ADU service to TES. Also the GDU service will be announced in the RUNNING state only.

The SAS may be set back to the RESET state via the

- RESET_APPLICATION

command.

To remove the SAS from memory the

- UNLOAD_APPLICATION

command must be given.
In case of error the SAS enters the state ABORTED and must the removed from the test node using the UNLOAD_APPLICATION command as well.

Note that the sequence of state transitions is a guide-line only.
The SAS programmer is free to modify the SAS reactions upon the commands.
Example: The command INIT_APPLICATION *is not mandatory and may be omitted, if the programmer decides to do the initialisation after the* LOAD_APPLICATION *command. Nevertheless the programmer must guarantee that sending the* INIT_APPLICATION *command does not cause a SAS crash.*
Note that the SAS behaviour must be well documented and known as well to the programmer developing APs or operational procedures !

The current SAS state can be monitored by the operator on the SAS status display or using the

- GET_APPLICATION_STATUS

command which returns the status of an application including error status and statistics.

Two UCL library procedures are available for the communication with SAS, they allow to read resp. to send a string from/to a SAS.
The UCL library command

- READ_MESSAGE_FROM_APPLICATION

additionally has a blocking indicator, which allows to specify if the procedure shall wait until an application really sends a message or if only the next message buffered is read.

- WRITE_MESSAGE_TO_APPLICATION

sends a string of up to 255 characters to the SAS.

Note that the contents of the messages to be exchanged depends on the implementation of the SAS!

Additionally the UCL library command

- DOWNLOAD

is provided. This commands downloads a file to a SAS, the corresponding frontend equipment managed by the SAS or the Unit Under Test (UUT) through a SAS.

7.3 Preparing Ground Synoptic Displays

The graphic display editor is implemented by the ground window definition utility (GWDU). The GWDU is capable of editing so called synoptic displays. Synoptic displays are graphical representations of the test object. The display is animated by data delivered from the test object (e.g. a sensor represented by a thermometer displayed in the synoptic)

The user should refer to the GWDU User Manual (see reference 2.1.2.1) for details on use of this tool.

The GWDU tool is started via the I_MDB Tools menu on end items of type WDU_GROUND_SYMBOL and WDU_GROUND_SYNOPTIC_DISPLAY, see Figure 7–28.

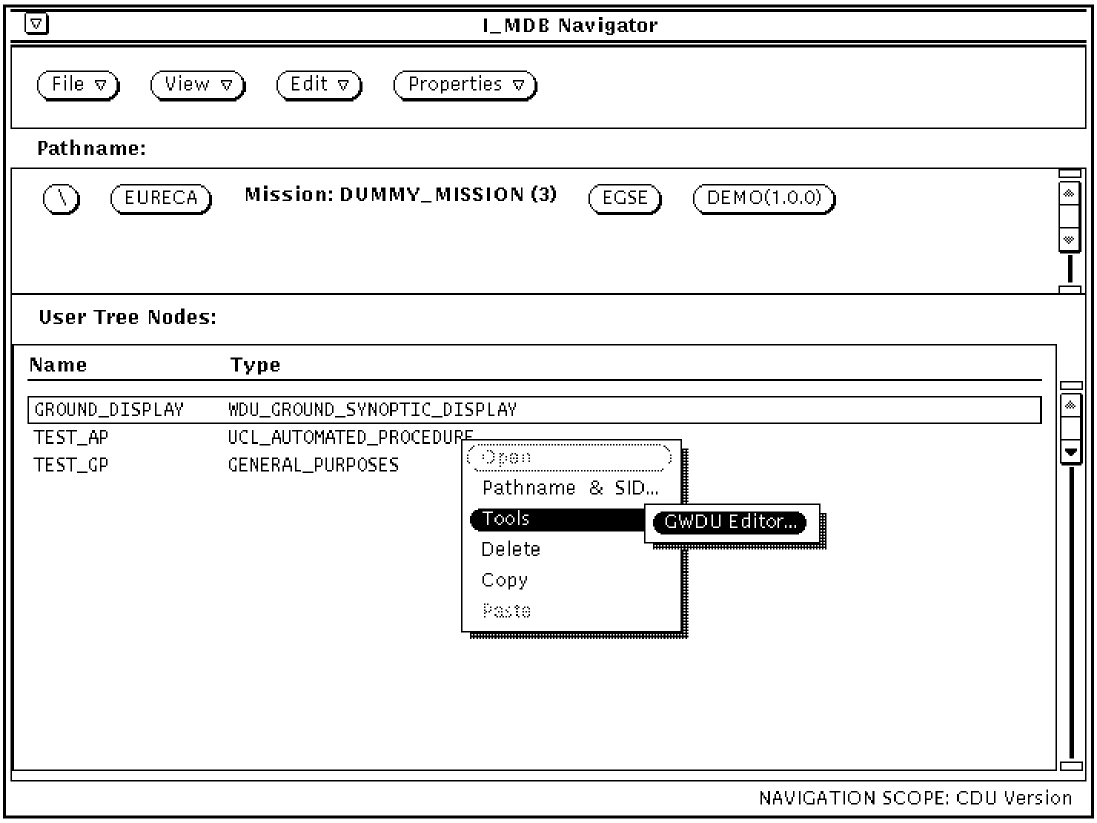


Figure 7–28 : *I_MDB provides the mechanism to start the GWDU Editor*

7.4 Developing Simulation Models

This chapter describes the graphical model development language MDE–GL which is used in the Model Development Environment (MDE) to implement simulation models. The aim is not only to describe the different language elements and their use but also to gain an insight into the timing concepts of model execution. The latter is needed to understand the concepts of modelling and to develop CSS models successfully.

7.4.1 MDE–GL Language Elements

Figure 7–29 presents the MDE–GL lexical elements.

7.4.1.1 Composite Function Blocks

Composite function blocks (CFBs) are implemented graphically by MDE–GL. The various MDE–GL lexical elements, specifically interconnected atomic function blocks, parameter blocks and composite function blocks may be composed to one composite function block. The interface items of a CFB, which are accessible from outside and inside the function block, allow to draw connections into and out of a CFB.

Since the implementation of a composite function block may consist, among others, of further composite function blocks, composite function blocks constitute the nodes in the hierarchical (simulation) model function block tree structure. The root of this tree structure is the single top level composite function block (TLFB), up to 16 levels of decomposition are possible.

The top level composite function block represents the (simulation) model source, it is named with the model's name and its interface represents the simulation model's (external) interface. The interface items may reference onboard items (i.e. stimuli and measurements) in the MDB allowing to specify the connection of the simulation model (the simulator kernel) to an external system.

7.4.1.2 Atomic Function Blocks

Atomic function blocks are implemented by AIL code, either directly or by a decision table which is automatically transformed into AIL code. Each atomic function block can be seen as an AIL procedure, the specification of which is automatically derived from the MDE–GL graphical definition of its input/output interface: the outputs denote the in–out parameters while the inputs denote the in parameters. The incoming connections (ending at an input) and outgoing connections (starting at an output) denote the data flow.

Usually the AIL implementation code computes new output values based on input values passed to the function block, but also (old) output values may be accessed. An output of an atomic function block may be connected (finally) to inputs of atomic function blocks as well as to model (i.e. TLFB) outputs. An input of an atomic function block may be connected (finally) to an output of an atomic function block, the output of a parameter block or a model (i.e. TLFB) input.

There are two different types of atomic function blocks: Synchronous atomic function blocks (SFBs) are activated periodically with the user specified time frame. Asynchronous atomic function blocks (AFBs) are activated by events, i.e. whenever an activation event occurs for at least one of its inputs (see section 7.4.2).

7.4.1.3 Parameter Blocks

A parameter block (PB) keeps one output. In contrast to atomic function blocks, parameter blocks are never executed and therefore the value of the output stays constant, unless it is changed by the user via an assignment.

7.4.1.4 Interface Items

Three kinds of interface items are provided: Inputs, outputs and grouping links.

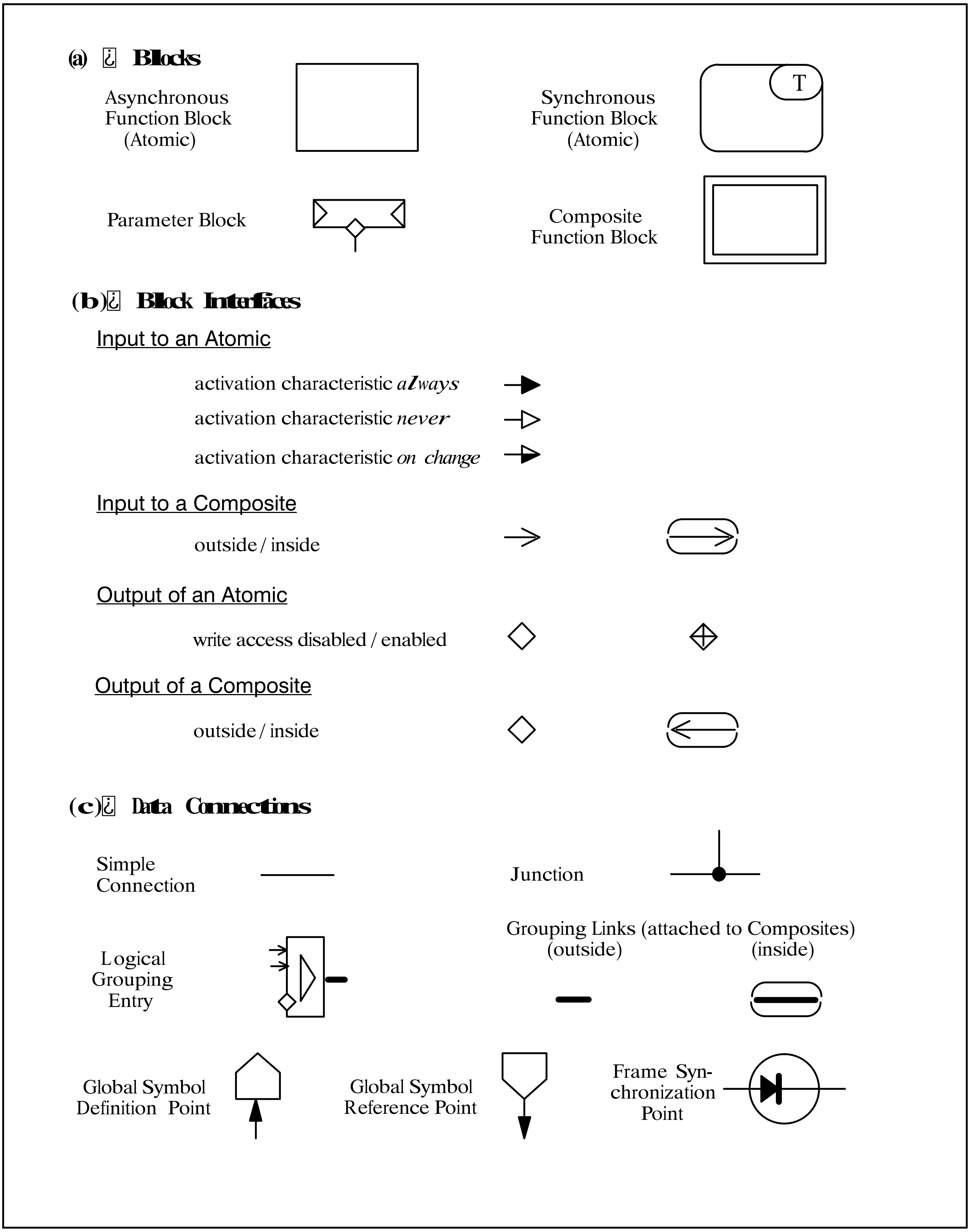


Figure 7–29 : *MDE Graphical Language Lexical Elements*

The inputs and outputs of an atomic function block denote the in and in–out parameters of the corresponding AIL procedure. The output of a parameter block represents a constant value. Inputs and outputs of a composite function block allow to draw connections into and out of the composite function block. The inputs and outputs of the top level composite function block (TLFB) denote the external interface of the simulation model. Inputs and outputs of grouping entries represent the signals contained in the logical grouping.

The outputs of atomic function blocks and parameter blocks and the model (TLFB) inputs represent the data sources, whereas the inputs of atomic function blocks and model (TLFB) outputs represent the data sinks. Each data sink must be connected to a data source.

Each input and output is associated with a particular data type chosen from a predefined set of data types. Connections between interface items are restricted to items of compatible type.

Inputs of atomic function blocks are marked with an activation characteristic. Outputs of atomic function blocks may be marked for write access (i.e. to enable the user to set the value of the output during simulation execution).

Grouping links may be attached to composite function blocks, allowing to draw logical groupings into and out of them.

7.4.1.5 Simple Connections

Simple connections allow to connect the various outputs and inputs (of atomic function blocks, composite function blocks, parameter blocks and grouping entries) in the MDE–GL implementation of a particular composite function block together or to global symbols, thereby specifying the data flow.

Also grouping entries may be connected together or to grouping links at the interface of composite function blocks.

7.4.1.6 Logical Groupings

A logical grouping is a bus of signals, each signal has a unique name within the grouping and can be used to connect interface items. A logical grouping is built by a set of interconnected grouping entries. Signals are accessed by the inputs and outputs of the grouping entries; each signal consists of one input at one of the grouping entries and of one or more outputs at other grouping entries connected. The input and the outputs representing a particular signal are named with the same name, i.e. the name of the signal; they can be connected to the outputs and inputs of function blocks, the outputs of parameter blocks or to global symbols. Grouping links allow to draw a logical grouping into and out of composite function blocks.

Grouping entries connected to the TLFB interface represent a special case: The number of inputs and outputs that can be attached directly to the interface of the TLFB is restricted by the extent of the graphic. In order to avoid any limitation on the number of inputs and outputs at the simulation model external (i.e. TLFB) interface, grouping entries may be connected to a grouping link at the interface of the TLFB. The outputs of these grouping entries represent model inputs, the inputs of these grouping entries represent model outputs. Here, the signal mechanism (i.e. one signal input, several signal outputs) is not supported, grouping entries connected to the TLFB interface may not be interconnected with ordinary logical groupings as described above.

7.4.1.7 Global Symbols

A global symbol provides another means for connecting interface items. It consists of one definition point and one or more reference points. Each reference point is connected to the definition point with the same name, found in the same composite function block as the reference point or in the next higher level composite

function block with respect to the line of CFBs, beginning with the CFB which contains the reference point straight up to the TLFB. The definition point and the reference points can be connected to the outputs and inputs of function blocks resp. grouping entries and to the outputs of parameter blocks.

7.4.1.8 Frame Synchronization Points

The purpose of the Frame Synchronization Point is todelay a signal's propagation to a receiver (usually an asynchronuous FB) until the start of the following simulation time frame.

Thus, FSPs are the tool to split asynchronuous chains and, especially, loops (see section 7.4.2.3).

Note: In the current implementation, inputs of AFBs with activation ***never*** are treated as if connected via an implicit FSP, i.e. they will receive changed values from their senders in the following simulation time frame.

Editorial Note:
Since the Frame Synchronization Point has been added only recently, most screen snapshots of the model editor window in this chapter have not been updated to include the corresponding icon.

7.4.2 Model Execution Strategy

The execution of the atomic function blocks is based on a time frame concept. The basic time frame is called the simulator minframe. All other time frames are multiples of the minframe.

7.4.2.1 Input Activation Modes

AFBs and hibernating SFBs have to be marked for activation in order to become executed. This is done when a specific activation event occurs for at least one of the FB's inputs. Three different input activation characteristics are provided:

- *always*
  The FB is marked for activation if the parent FB of the output connected to the input has been executed; if the input is connected to a model input, it is marked for activation if the referenced stimulus has been written by the external system.
  In addition, the FB is marked for activation, if an assignment issued by the user to the output resp. model input connected to the input or to the input itself has occured.
- *on change*
  The FB is marked for activation if the value of the output resp. model input connected to the input has changed, i.e. has been updated with a value different from its value before the update. Such an update may result from a calculation of an output during execution of the output's parent FB, from a write operation of the external system to a stimulus referenced by a model input or from an assignment issued by the user.
  In addition, the FB is marked for activation, if an assignment issued by the user to the input itself, that changed the input's value, has occured.
- *never*
  Such an input has no impact on activation of its parent FB.

Note that the occurence of such an event does not lead to an immediate execution of the FB, instead the FB is marked for activation and will be executed synchronized with its specific time frame. Also, the occurence of multiple events before the FB is actually executed does not lead to multiple execution, a number of events greater than 1 is handled the same way as just 1 event.

Inserting a FSP into a signal will delay not only thecorresponding value update, but also the activation for a receiving AFB until the following simulation time frame. If the AFB also has non–synchronized inputs, this means that it may be activated twice in two consecutive frames.

A special semantic is associated with the pulse types (i.e. PULSE resp. BURST_PULSE). An update of an interface item of a pulse type (resulting from a calculation of an output during execution of the output's parent FB, from a write operation of the external system to a stimulus referenced by a model input or from an assignment issued by the user) is referred to as a trigger. This indicates that, in contrast to all other data types, the value does not persist until the next update, for each AFB or SFB input it is reset automatically when the execution of the parent FB has been completed. Therefore each update (trigger), even consecutively with the same value, marks all connected FBs (in the case of a trigger of a model input resp. of an AFB, SFB or PB output) resp. the parent FB (in the case of a trigger of an AFB or SFB input) for activation. AFB and SFB inputs of pulse type must be marked with the activation characteristic *on change*.

7.4.2.2 Synchronous Function Blocks

Unless hibernating, SFBs are executed periodically with a time frame specified as a multiple of the simulator minframe. Each execution period lasts the specific SFB's time frame. The function code may include an AIL statement to suspend the cyclic activation temporarily, causing the SFB to hibernate. A hibernating SFB has to be marked for activation before it is executed periodically again.

An SFB is executed exactly once per execution period. For each SFB, the exact start and end times of its execution periods with respect to the start of the simulation, i.e. the first minframe (minframe 1) are fixed for the entire simulation: The n–th execution period starts with the beginning of the ((n–1) * T + 1)–th minframe and ends with the (n * T)–th minframe where T is the SFB's time frame (duration of the execution period) in multiples of the minframe. The first execution period of each SFB starts with the beginning of the simulation, i.e. with the beginning of the first minframe. During execution, an SFB may set itself hibernating, this means that it will be suspended temporarily for execution (in its following execution periods) as long as it is not marked for activation (see 7.4.3.1). Suspension and reactivation of an SFB are synchronized with its time frame (i.e. they have no impact on the execution period scheme described).

The value of an output of an SFB becomes available (to connected inputs of SFBs, to connected inputs of AFBs and to connected model (TLFB) outputs) at the end of the SFB's execution period.

If the activation criteria are met, hibernating SFBs connected to an output of an SFB are marked for activation at the end of the SFB's execution period, they will be executed in their usual next execution period. AFBs connected to an output of an SFB are marked for activation also at the end of the SFB's execution period, they will be executed in the following minframe.

For each execution, the inputs of an SFB are parameterized with the values which are available at the beginning of the SFB's execution period; outputs of an SFB retain their last computed value. Since SFBs operate on input value snapshots made at the beginning of the execution period, they can be seen as parallel processes that are independent from activation orders.

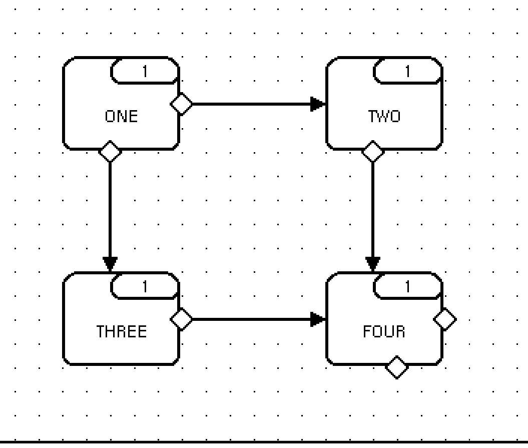


Figure 7–30 *: A number of SFBs connected together*

Figure 7–30 shows some connected synchronous function blocks. All function blocks are executed *concurrently* within their time frame.

Synchronous function blocks may be used to trigger the execution of asynchronous groups (see next section).

7.4.2.3 Asynchronous Function Blocks

AFBs connected together in any direction (i.e. not restricted to forward direction with respect to the arrow symbols of interface items) via at least one input of activation characteristic *always* and/or *on change* belong to the same asynchronous group. The AFBs in a group are serialized for execution, the order is defined by the connections between the interface items. A correct serialization is any total ordering of the group member FBs which complies with the following rule:

> Any group member FB_i is executed after all group members FB_j with a connection from an output of FB_j to an input with activation characteristic *always* or *on change* of FB_i (a necessary and sufficient precondition is the acyclicity of the asynchronous group).

Depending on the situation more than one order may be legal. CSS transforms the graphical model specification into a linear program structure, automatically choosing one of several possibilities, if necessary.

Each asynchronous group is processed as a whole during one minframe. Processing means that, one by one in a correct order, each AFB that is marked for activation is executed (each AFB marked for activation is executed exactly once, the execution of an AFB may mark a succeeding AFB of the same group for activation); AFBs not marked for activation are skipped.

The value of an output of an AFB becomes available a) to connected inputs of SFBs, to connected inputs of AFBs of activation characteristic *never* and to connected model (TLFB) outputs at the end of the minframe the AFB is currently executed in and b) to connected inputs of AFBs of activation characteristic *always* resp. *on change* immediately, i.e. in the current minframe, unless the connection is interrupted by a FSP, in which case the value becomes available in the following minframe.

If the activation criteria are met, hibernating SFBs connected to an output of an AFB are marked for activation at the end of the minframe the AFB is currently executed in, they will be executed in their usual next execution period. AFBs connected to an output of an AFB are marked for activation immediately, they will be executed in the current minframe.

For each execution, the inputs of an AFB are parameterized with the values which are available at the time of the beginning of the AFB's execution; outputs of an AFB retain their last computed value.

Figure 7–31 shows an asynchronous group triggered by an SFB running every minframe. During the first minframe none of the AFBs is executed (no computations are performed) because none of the AFBs has been marked for activation at this time. At the end of the first minframe, the AFB ONE is marked for activation by SYNC. During the second minframe the complete asynchronous group will be processed (when the asynchronous group of this example is processed, always all AFBs are executed because all inputs are of activation characteristic *always*).

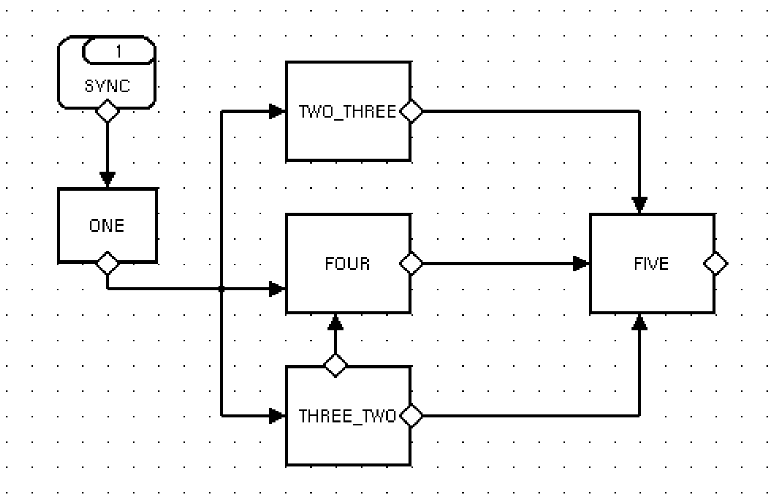


Figure 7–31 : *An example of an asynchronous group triggered by a synchronous function block*

The graphical specification allows 2 serializations, both are legal:

SYNC: –> ONE –> TWO_THREE –> THREE_TWO –> FOUR –> FIVE or
SYNC: –> ONE –> THREE_TWO –> TWO_THREE –> FOUR –> FIVE

Note that the AFB FOUR will be executed once (with the relevant values from ONE and THREE_TWO) although it is marked for activation twice (arrows from FB ONE and FB THREE_TWO).

The user can not determine whether the function block TWO_THREE or the function block THREE_TWO will be executed first.

The situation becomes more complicated if the user changes some of the input activation characteristics to *on change* (the parent FB will be activated if the predecessor FB connected to the input has changed the value of the output connected to the input). Both serializations are still valid but it is possible that the execution of single function blocks is skipped.

Figure 7–32 shows a case where two SFBs of different time frame are used as triggers in order to reduce computation load during model execution.

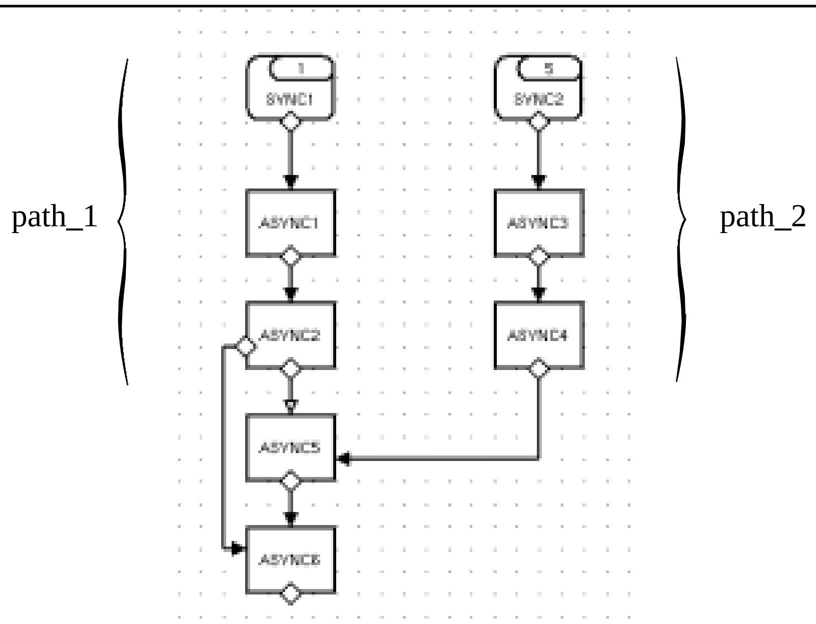


Figure 7–32 : *Use of two synchronous function blocks to reduce system load*

The AFBs all belong to the same asynchronous group, but the AFBs in path_1 and also ASYNC6 will be executed every minframe (the SFB SYNC1 is running every minframe) whereas the AFBs in path_2 and also ASYNC5 will be executed every fifth minframe. Note that if the connection between ASYNC2 and ASYNC6 would be missing or the input of ASYNC6 connected to the output of ASYNC2 would be of activation characteristic *never*, there would be two different asynchronous groups (ASYNC1–ASYNC2 and ASYNC3–ASYNC4–ASYNC5–ASYNC6).

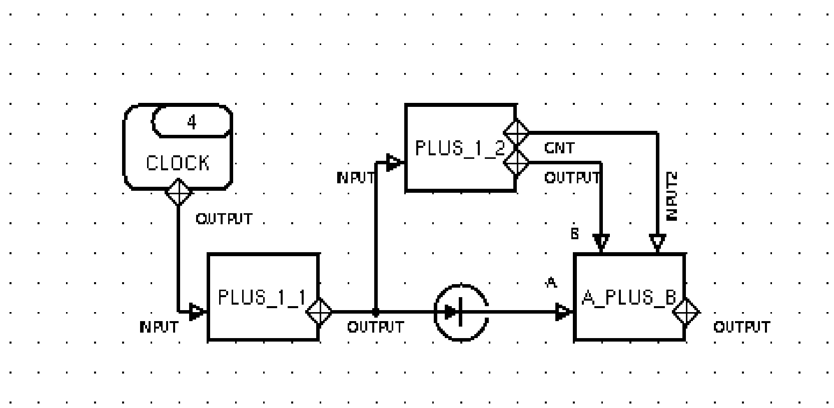


Figure 7–33 : *Asynchronuous chain with FSP and activation* ***never***

In Figure 7–33, we finally see an asynchronuous chain with FSP and one input with activation mode ***never.*** Note that this is just an example for explanation of the activation mechanisms, not for good modelling style!

The output of SFB CLOCK triggers AFB PLUS_1_1 every 4 minframes. This will immediately activate AFB PLUS_1_2 (via INPUT), followed by A_PLUS_B (via B, assuming their input values have changed). Due to the FSP connected to input A and mode ***never*** (with implicit FSP) of input INPUT_2, A_PLUS_B will receive the just calculated value of OUTPUT from PLUS_1_2 as input B, but the previous values of OUTPUT from PLUS_1_1 as A and of CNT from PLUS_1_2 as INPUT_2.

In the following minfarme, due to the FSP, the delayed update of OUTPUT from PLUS_1_1 will activate A_PLUS_B once again, now with the updated values for A and INPUT_2 from the previous minframe.

7.4.2.4 Connection to an external system (H/W in the loop)

If a simulation model (a simulator kernel) is connected to an external system (which sends stimuli to and receives measurements from the model) its timing behaviour is still determined by the frame machine concept. Model inputs are treated in the same way as outputs, model outputs are treated in the same way as inputs of synchronous function blocks with an execution period of 1 minframe.

The minimum model response time resulting from this timing behaviour is 2 minframes (of course, if the implementation of a simulation model requires multiple minframes to perform the computations of the model outputs based on values passed for the model inputs by the external system, the model response time increases accordingly).

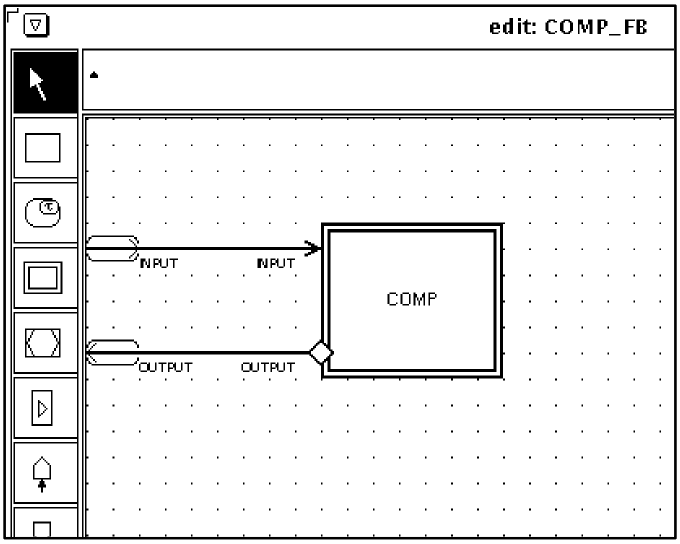


Figure 7–34 : *A model with external interface (one model input and one model output).*

Another consequence of the described timing mechanism is that only the last of a series of multiple events occurring within one execution period will take effect – all previous events will be lost. This may happen if events generated by the external system are processed by model functions executed with a slower frequency than the external event frequency.

7.4.2.5 The Simulation State

The simulation state contains the complete dynamic state information of a simulation model at a given time during a simulation execution, it comprises the current values of all atomic function block outputs, parameter block outputs and model inputs, the current values of all atomic function block inputs and model outputs as well as the current function block activation states.

When a simulator kernel is configured, a simulation state that covers the initial model state information is automatically generated. It is based on the initial values of the atomic function block outputs, parameter block outputs and model inputs and the initial running/hibernating states of synchronous function blocks as entered by the model developer using the MDE tools.

7.4.3 Implementation Of Atomic Functions

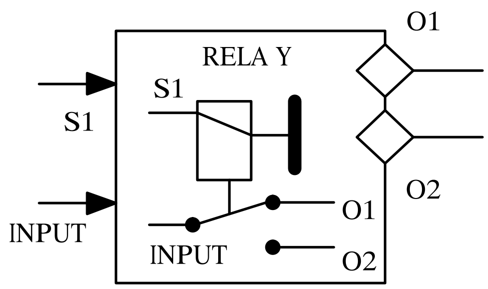
As mentioned before, atomic functions must be implemented either by decision tables or using AIL code (AIL – Atomic Implementation Language, defined as an Ada subset). AIL covers the elementary Ada features necessary for the implementation of atomic function blocks, as arithmetic and logic operations, conditional branching and loops.

Certain Ada features are explicitly forbidden for the implementation of atomic functions to avoid conflicts with the simulator kernel. In practice, this means that certain Ada keywords (see 7.4.3.1) are rejected by the Atomic Implementation Editor, and that the usual Ada standard library packages are inaccesible. The forbidden Ada features (e.g. tasking or standard I/O packages) are not needed for the implementation of atomic functions: All necessary system programming functions for parallelization, synchronization, function activation etc. are completely specified by the MDE–GL model definition.

Local variables may be declared in which case their lifetime is limited by one execution of the associated function (i.e. there are no static variables). Furthermore, local subprograms may be declared and defined. No object declared inside the code of one function block can be made visible to another function block (i.e. there are no global variables), and no function block implementation can call the code of any other function block directly. This ensures that the MDE–GL graphical connections are the only means of communication between model functions.

Each atomic function block can be seen as an Ada procedure, the specification of which is automatically derived from the MDE–GL graphical definition of its input/output interface (cf. Figure 7–35).

GRAPHICAL REPRESENTATION IMPLEMENTATION




```
procedure RELAY (
    S1     : in STATE_CODE;
    INPUT  : in LONG_REAL;
    O1 : in out LONG_REAL;
    O2 : in out LONG_REAL) is
begin
   if S1 = ”OFF     ”
      then O1 := INPUT;  O2 := 0.0;
      else O1 := 0.0;    O2 := INPUT;
   end if;
end RELAY;
```

underlaid code is generated automatically

Figure 7–35 : *MDE–GL Atomic Function Block with AIL Implementation*

The parameters to such a procedure are the following:

– one (Ada–) in parameter for each function block input, initialized with the value which is available at the beginning of the AFB execution resp. SFB execution period

– one (Ada–) in out parameter for each function block output, initialized with the current value of this output variable (i.e. the value computed during the last execution of this function, or the initial value when the function is activated for the first time)

7.4.3.1 Description of AIL

The Ada subset covered by AIL encompasses:

– the declaration of types (excluding task– and access types), constants and variables
– standard arithmetic and logical operations
– standard control structures, such as branches and loops
– the definition of subprograms (functions and procedures)
– the inheritance of certain standard library functions, e.g. math functions

Figure 7–35 illustrates the use of AIL for the implementation of atomic function blocks and its automatic integration with graphical MDE–GL function block definitions.

Identifiers

The syntax of AIL identifiers is equal to Ada identifier syntax, i.e. identifiers may consist of letters (a..z, A..Z), digits (1..9) and the underline character. An identifier must start with a letter and may not end with an underline; the consecution of two underlines is ruled out.

Predefined Visible Declarations Common To All Atomic FB Implementations

The following standard and CSS specific Ada declarations are implicitly visible to the atomic FB implementation code (AIL or decision table conditions/actions). At the same time, they are the *only* visible declarations:

Standard (Predefined) Types; Literals

The following standard data types as defined by the Ada LRM (3.5.2–3.5.4, 3.5.7, 3.6.3) are provided:

```
BOOLEAN   INTEGER   FLOAT   CHARACTER   STRING
```

Literals of these types are written as described in the Ada LRM 2.4–2.6.

Examples:

```
TRUE            FALSE                          -- boolean literals

0               +20            –14362          -- integer literals
0.0             +20.0          –0.123          -- real literals
1.23E–12        +1.0E+6        –4.67E–2        -- real literal with exponent

2#1111_1111#    16#FF#         016#0ff#        -- literal for 255, expressed
                                               -- for different bases

’a’             ’X’            ’@’             -- character literals
”hello”         ”Hi there!”    ”!@#$%^&*”      -- string literals
```

CSS Specific (Predefined) Types

CSS provides the following additional types to be used as types for function block inputs/outputs and for the declaration of local variables. However, not all of these types may be assigned to the inputs/outputs of a simulation model (i.e. top level composite function block); in the following listing the types that may be assigned also to model interface items are underlined. The types are contained in the Ada package CSS_TYPES, the declarations are implicitely visible to atomic function block implementation code (note that the type BOOLEAN, although a standard predefined type, is listed here again for the reason of completeness, since it may be assigned to function block inputs/outputs):

– <u>UNSIGNED_BYTE</u>
is an integer type with the range 0..+2**8–1.

– SIGNED_BYTE
is an integer type with the range –2**7..+2**7–1.

– UNSIGNED_SHORT_WORD
is an integer type with the range 0..+2**16–1.

– SIGNED_SHORT_WORD
is an integer type with the range –2**15..+2**15–1.

– UNSIGNED_INTEGER
is an integer type with the range 0..+2**31–1 (note that 2****31**–1 is not a typographical error).

– SIGNED_INTEGER
is an integer type with the range –2**31..+2**31–1.

– REAL
is a floating point type in 32 bit IEEE format (IEEE single float). To inputs and outputs an engineering unit can be assigned; type compatibility checking between inputs and outputs is extended to identical engineering unit.

– LONG_REAL
is a floating point type in 64 bit IEEE format (IEEE double float). To inputs and outputs an engineering unit can be assigned; type compatibility checking between inputs and outputs is extended to identical engineering unit.

– BOOLEAN
is actually a standard predefined type.

– STATE_CODE
is an ”enumeration type” with user selectable representation; type compatibility checking between inputs and outputs is extended to identical representation. Actually it is a string type of 8 characters, each string must comply with AIL identifier syntax, possibly filled up with trailing blanks (<space> signs).

Example:

```
OUTPUT := ”ON      ”;   –– statecodes: ”OFF     ”, ”ON      ”
```

– TIME
is a private type the internal representation of which is hidden; there are no literals. The following functions and procedures may be used to assign resp. access variables:

```
subtype YEAR_NUMBER is INTEGER range 1901..2099;
subtype MONTH_NUMBER is INTEGER range 1..12;
subtype DAY_NUMBER is INTEGER range 1..31;
subtype DAY_DURATION is LONG_DURATION range 0..86400.0;

function TIME_OF (YEAR: YEAR_NUMBER;
     MONTH: MONTH_NUMBER;
     DAY: DAY_NUMBER;
     SECONDS: DAY_DURATION := 0.0) return TIME;
function YEAR (DATE: TIME) return YEAR_NUMBER;
function MONTH (DATE: TIME) return MONTH_NUMBER;
function DAY (DATE: TIME) return DAY_NUMBER;
function SECONDS (DATE: TIME) return DAY_DURATION;
procedure SPLIT (DATE: in TIME;
     YEAR: out YEAR_NUMBER;
     MONTH: out MONTH_NUMBER;
     DAY: out DAY_NUMBER;
     SECONDS: out DAY_DURATION);
```

TIME includes a function CLOCK which returns the current simulated mission time (SMT). Additionally a function GET_LOCAL_TIME is provided which returns the current local time.

Examples:

```
OUTPUT := CLOCK;                    -- current SMT

OUTPUT := GET_LOCAL_TIME;           -- current local time

OUTPUT := TIME_OF (1997, 12, 24);-- december the 24th, 1997
```

– LONG_DURATION
is a fixed point type with the range –86.400.0..+86.400.0 seconds. It includes a function GET_MINFRAME_INTERVAL which returns the increment in simulated mission time (SMT) per minframe. The syntax for literals is equal to the syntax of REAL literals.

– COMPLEX
is a record type. The two components RE and IM, both of type LONG_REAL, denote the real and imaginary part of a complex number in Gauss notation.

Examples:

```
OUTPUT := (10.0, –5.0);      -- set real part to 10.0, imag. part to –5.0

OUTPUT.RE := 10.0;           -- set real part to 10.0
OUTPUT.IM := –5.0;           -- set imaginary part to –5.0
```

– PULSE
allows to trigger a pulse output (on the sender side) and to test if a given pulse input was triggered. It provides a simple trigger mechanism for cases in which only event propagation, not a concrete value is relevant. The predefined constant PULSE_TRIGGERED is provided for triggering (assignment) and testing a variable in AIL code.

Examples:

```
OUTPUT := PULSE_TRIGGERED;                               -- trigger
if (INPUT = PULSE_TRIGGERED) then ... end if;            -- test
```

– BURST_PULSE
allows to trigger a burst pulse output (on the sender side) and to test if a given burst pulse input was triggered. It provides a trigger mechanism that comprises the transfer of information, i.e. the number of pulses. The predefined constant BURST_PULSE_NOT_TRIGGERED is provided for testing a variable in AIL code. The syntax for literals is equal to the syntax of UNSIGNED_INTEGER literals.

Example:

```
OUTPUT := 3;                                             -- trigger
if (INPUT /= BURST_PULSE_NOT_TRIGGERED) then             -- test
     if INPUT > 5 then ... end if;
end if;
```

– Various VECTOR types:
UNSIGNED_BYTE_VECTOR, SIGNED_BYTE_VECTOR,
UNSIGNED_SHORT_WORD_VECTOR, SIGNED_SHORT_WORD_VECTOR,

UNSIGNED_INTEGER_VECTOR, SIGNED_INTEGER_VECTOR,
REAL_VECTOR, LONG_REAL_VECTOR,
BOOLEAN_VECTOR,
COMPLEX_VECTOR
Inputs/outputs of VECTOR types must consist of at least 2 elements, the maximum VECTOR size is restricted to 255 elements.

Examples:

```
OUTPUT := (8.1, –17.7, 4.5);-- REAL_VECTOR of 3 elements
OUTPUT(1) := 8.1;
OUTPUT(2) := –17.7;

X: BOOLEAN := INPUT(5);      -- access 5th element of a BOOLEAN_VECTOR
```

– Various MATRIX types:
UNSIGNED_BYTE_MATRIX, SIGNED_BYTE_MATRIX,
UNSIGNED_SHORT_WORD_MATRIX, SIGNED_SHORT_WORD_MATRIX,
UNSIGNED_INTEGER_MATRIX, SIGNED_INTEGER_MATRIX,
REAL_MATRIX, LONG_REAL_MATRIX,
BOOLEAN_MATRIX,
COMPLEX_MATRIX
Inputs/outputs of MATRIX types must consist of at least 2 columns and 2 rows (i.e. 4 elements), the maximum MATRIX size is restricted to 255 (i.e. columns * rows) elements.

Examples:

```
OUTPUT := ((7.3, –2.5, 4.0), (1.5, 3.9, –2.7));-- REAL_MATRIX of 2 rows
OUTPUT(1,1) := 7.3;                            -- and 3 columns
OUTPUT(2,3) := –2.7;

X: BOOLEAN := INPUT(5,3);             -- access element in 5th row and
                                      -- 3rd column of a BOOLEAN_MATRIX
```

– RECORD
The RECORD type provided for function block outputs and inputs allows to simplify the MDE–GL graphics: Instead of creating multiple outputs at one atomic FB and the corresponding number of inputs at another atomic FB and connecting each of them, it is possible to create one output and one input of RECORD type and to connect them by a single connection. This RECORD type is not available in the AIL implementation code; each component of such an output/input is treated as an individual scalar output/input, accessed by its name which consists of the concatenation of the output resp. input name, the underline character and the component name. Such a RECORD may consist of the following scalar types:
UNSIGNED_BYTE, SIGNED_BYTE,
UNSIGNED_SHORT_WORD, SIGNED_SHORT_WORD,
UNSIGNED_INTEGER, SIGNED_INTEGER,
REAL, LONG_REAL,
BOOLEAN,
COMPLEX

Example:

```
OUTPUT_X := 2.3                   -- an output named OUTPUT of RECORD
OUTPUT_Y := 7;                    -- type consisting of three components:
OUTPUT_Z := TRUE;                 -- X (REAL), Y (UNSIGNED_INTEGER) and
                                  -- Z (BOOLEAN)
```

User Defined (Local) Types

The user may define additional types, e.g. subtypes of predefined types (subtypes are useful if the user’s abstraction of the problem space defines subsets of a base type, as e.g. a range constraint of allowed temperature values) or new enumeration types, to improve the readability of the atomic implementation code. Also array and record types may be declared. The visibility of such type declarations is limited to the scope of a single atomic function block within which they are declared (i.e., user defined types are strictly local to a function block). The syntax of these types is defined by the Ada LRM (3.3.2, 3.5.1, 3.6, 3.7).

Examples:

```
type temperature  is range 1.0 .. 2_000.0; -- a type declaration
subtype SMALL_INT is INTEGER range –10 .. 10;
type RELAY_POSITION  is (OFF, ON);         -- an enumeration type declaration
type TABLE is array (1..10) of INTEGER;    -- a simple array type declaration
type BUFFER is                             -- a simple record type declaration
     record
          POS: INTEGER := 0;
          VALUE: STRING (1..100);
     end record;
```

Since user defined types are always declared locally inside the implementation code of an atomic function block, only predefined types may be used for a block’s inputs or outputs.

Local Constant and Variable Declarations

Local constants and variables of any standard, predefined or user defined type may be declared, invisible outside the function block in which they are declared, and with a lifetime limited by one execution of the corresponding function.

Examples:

```
PI        : constant := 3.1459_26536;      -- a real number
ONE, EINS : constant := 1;                 -- two different names for 1
MESSAGE   : constant STRING
               := ”ERROR IN FUNCTION F1”;  -- a constant string
COUNT, SUM    : INTEGER;                   -- a variable declaration
```

Expressions

Expressions may be formed as defined by the Ada LRM (chapter 4, excluding 4.8). The following examples are legal AIL expressions (provided that all shown variables are declared to be of appropriate types):

Examples:

```
4.0                       -- real literal
SUM                       -- variable
INTEGER'LAST              -- attribute
SINE(X)                   -- function call
COLOR'(BLUE)              -- qualified expression
(LINE_COUNT + 10)         -- parenthesized expression
-4.0 + A                  -- compound expression
B**2 - 4.0*A*C            -- compound expression
X = 3                     -- relation
COUNT in SMALL_INT        -- relation
```

Statements

An AIL statement may be one of the following:

– the `null` statement

– a simple statement:
an assignment, `return`– or `exit` statement, or a procedure call

– a block:
non–empty list of statements (each followed by ';') enclosed in
`begin ... end`, resp. `declare ... begin ... end`

– a conditional statement:

```
if ... then ... end if
if ... then ... else ... end if
if ... then ... elsif ... end if
case ... is when ... => ... when others => ... end case
```

– a loop statement:

```
loop ... end loop
while ... loop ... end loop
for ... in ... loop ... end loop
for ... in reverse ... loop ... end loop
```

Syntax and semantics of these statements are as defined by the Ada LRM (chapter 5).

Examples:

```
OUT1 := COS(PI);                -- assignment statement

if IN1 < 0 then OUT1 := 5;      -- if statement
else OUT1 := 6;
end if;
```

User Defined (Local) Subprograms

The user may define local functions and procedures. These can be called only from within the code of the function block they are defined in. The syntax for functions and procedures is described in the Ada LRM (chapter 6).

Examples:

```
function MAX (X, Y: LONG_REAL) return LONG_REAL is
begin
     if X > Y then
          return X:
     else
          return Y;
     end if;
end MAX;
```

```
procedure SWAP (X: in out REAL; Y: in out REAL) is
declare
     T: REAL;
begin
     T := X;
     X := Y;
     Y := T;
end SWAP;
```

Math Package

The Ada package CSS_MATH provides a variety of reliable and reusable mathematical subprograms. In addition it defines data types, and numerical and physical constants, see section 7.4.12. The declarations are implicitely visible to atomic function block implementation code.

Hibernating Synchronous Function Blocks

Each synchronous function block implicitely has a variable named HIBERNATE of type BOOLEAN. The function block can set itself into a hibernating state (i.e. it will be suspended temporarily for execution starting with the following execution period until it is reactivated by an activation event) by assigning the value TRUE to this variable in the AIL code. Note that the current execution period is always completed, i.e. statements eventually following the hibernating statement are still performed.

Example:

```
if ... then
     HIBERNATE := TRUE;                  –– start hibernating
end if;
```

Message Output

Two procedures are provided allowing to output messages from AIL. The first procedure is specified as follows:

```
type MESSAGE_TYPE is
     (INFO_MESSAGE, WARNING_MESSAGE, NON_CRITICAL_ERROR, CRITICAL_ERROR);
procedure ERROR_MESSAGE (MESSAGE: STRING;
     FUNCTION_BLOCK_NAME: STRING;
     TYPE_OF_MESSAGE: MESSAGE_TYPE := NON_CRITICAL_ERROR);
```

It allows to write a message (parameter MESSAGE) to the MOCS console window. The parameter FUNCTION_BLOCK_NAME should be set to the name of the atomic function block producing that message.

The second procedure makes use of the message file containing all the messages used by the simulator kernel and is specified as follows:

```
procedure ERROR_MESSAGE (MESSAGE_NUMBER: INTEGER;
     REPLACE_STRING: STRING;
     FUNCTION_BLOCK_NAME: STRING);
```

It allows to write a message to the MOCS console window, to the log file and to the Test Result Data Base (TRDB). If CSS has been started from HCI, the message can also be sent to the CGS Error Services (i.e. the CGS Message Handler window), depending on a flag associated with the message in the message file. The parameter MESSAGE_NUMBER identifies the message in the file. The parameter REPLACE_STRING contains the values to replace the place holders in the message, if there are any. If there is more than one place holder to be replaced, the values have to be separated by the <tab> sign (i.e. ASCII.HT) or the '@' character in the replace string. The parameter FUNCTION_BLOCK_NAME should be set to the name of the atomic function block producing that message.

The name of the message file is specified by the environment variable CSS_KERNEL_MESSAGES. The contents of this file is split into 5 sections labeled by keywords. One of these sections comprises messages to be sent from AIL, it is labeled by the keyword $CSS_AIL_MESSAGES. Here, the user can define messages in the following syntax:

No<tab>*CGSI*<tab>*Crit.*<tab>*Message*

No is the integer index allowing the parameter MESSAGE_NUMBER to address the respective message. The *CGSI* flag may be either Y or N indicating whether the message shall also be sent to the CGS Error Services (Y means yes, N means no). However, this directive is performed only if CSS has been started from HCI.
Crit. may be either I, W, N or C indicating the criticality of the message (INFO_MESSAGE, WARNING_MESSAGE, NON_CRITICAL_ERROR, CRITICAL_ERROR). Within the CGS Error Handler window, I and W are transfomed to ADVISORY, N corresponds to ORDINARY and C corresponds to SEVERE.
Message is the message string, it may contain place holders *%V1%*, *%V2%*, etc. that are replaced by the values in the parameter REPLACE_STRING.

The following example results in the message `This is an example!!!` that is written to the MOCS console window, to the log file, the TRDB, and, if possible, (i.e. if CSS has been started from HCI) is sent also to the CGS Error Services.

Example:

```
# entry in message file:
6913 Y    I    This %V2% an %V1%!!!

–– corresponding AIL call in Asynchronous FB \DEMO\RELAIS_2:
ERROR_MESSAGE (6913, ”example” & ASCII.HT & ”is”, ”\DEMO\RELAIS_2”);
```

Forbidden Ada Features

All of the following Ada keywords are forbidden (thereby eliminating the features associated with these keywords):

```
abort, accept, access, at, delay, entry, generic, pragma, select, separate,
task, terminate, with
```

This means, that e.g. address, length, and representation clauses are ruled out, as well as the unportable and unsafe use of pragmas. The MDE Atomic Function Editor will reject any use of the above keywords inside atomic implementation code.

Since only those declarations which are made implicitly visible (standard and predefined, see above) are known to the implementation code, the normal Ada standard library units such as SYSTEM, TEXT_IO, UN-

CHECKED_CONVERSION or UNCHECKED_DEALLOCATION are completely inaccessible (deliberately, to avoid conflicts with the simulator kernel).

Atomic Function Implementation Rules

The following rules summarize the constraints for AIL programming, which are checked in part by the MDE rule checking function and in part by the Ada compiler system. They apply to functions implemented completely using AIL code as well as to the definition of decision table conditions and actions:

- Atomic FB implementation code must not contain Ada keywords which are related to tasking, dynamic memory, exception handling, and type representation.
- The use of input variables (formal parameters) and output variables inside the implementation code of atomic function blocks must match their specified type.
- System calls are not allowed for the implementation of atomics.

7.4.3.2 Atomic Implementation by Decision Tables

For decision tables a table editor is provided that ensures the completeness and consistency of the table by construction.

A decision table consists of two parts: a set of conditions and a set of actions. A condition is an expression of type BOOLEAN (i.e. two–valued).

Conditions and actions are linked by a decision matrix which specifies a column for each possible combination of conditions. To reduce the table size, *‘no matter’* values for conditions may be used. Further, a column labelled *‘others’* is automatically generated to catch all combinations of conditions which are not explicitly specified. A second part of the matrix associates each column with a (possibly empty) set of action(s) to be executed whenever the corresponding combination of conditions is met.

An action is either (a) an executable AIL statement or (b) an assignment. In case (a), the statement is written in the action field, and all table columns for which this action shall be executed must be marked. In case (b), the action field of the table contains the name of a function block output, while the value to be assigned must be specified in the respective table column. If for an action the table column for a specific combination of conditions is left blank, the corresponding action is not executed (a) resp. no assignement is made (b).

All conditions are evaluated before any actions are executed. With the exception of this restriction, the evaluation order of conditions and the execution order of actions is arbitrary.

Code entered for conditions and actions must be correct AIL resp. Ada code with the restrictions of section 7.4.3.1.

A sketch of a decision table shown by the Decision Table Editor is given in Figure 7–36. The user may edit the shown decision table, which is initially empty. The Decision Table Editor will ensure by construction that the decision table is complete and consistent at all times.

The user may define local variables and macros to abbreviate expressions and statements. Local variables are always computed at the beginning of the function block execution, they may be read accessed in conditions and statements. A macro allows simple textual substitution. The syntax is:

name=substitution_text

The symbol *name* is replaced in conditions and actions by the arbitrary text *substitution_text*. The substitution text is the rest of the line following the equal sign, macro definitions over multiple lines are not possible. The macro expansions are performed in the order the macros are specified, i.e. it is possible to use macros in substitution texts, see the example.

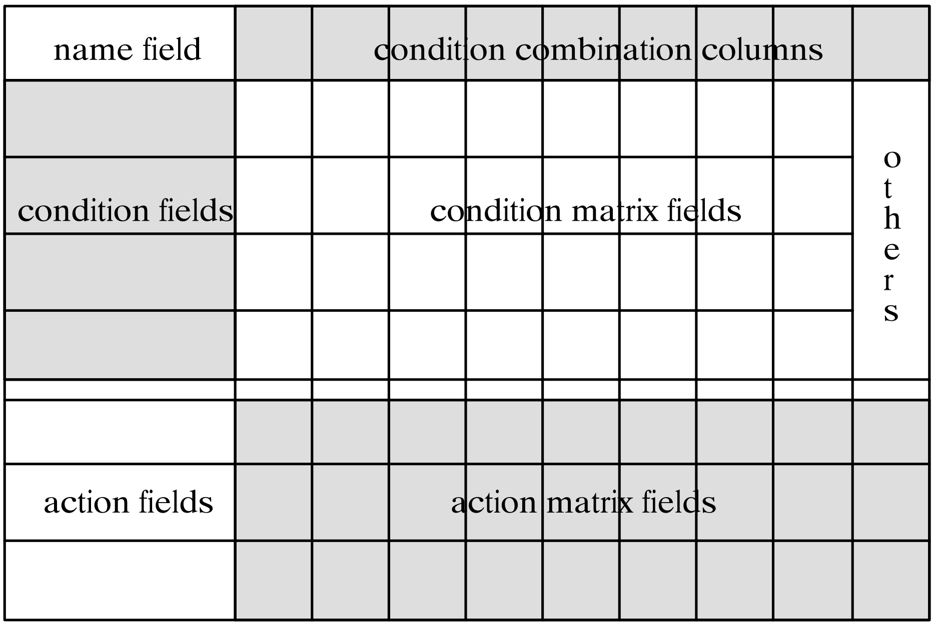


Figure 7–36 : *Sample outline of a decision table*

Examples:

```
MUL3=MUL2 * INPUT_3
MUL2=INPUT_1 * INPUT_2
```

The number of conditions is restricted to 8; otherwise a decision table may become too large to be handled easily.

Atomic Function Implementation Rules

The following rules summarize the constraints for AIL programming, which are checked in part by the MDE rule checking function and in part by the Ada compiler system. They apply to the definition of decision table conditions and actions:

- Decision table conditions must represent valid AIL resp. Ada expressions.
- Decision table actions must represent syntactically correct AIL/Ada statements.

7.4.4 Model Development Pre–requisites

Within I_MDB the creation and modification of nodes within the element configuration tree shall not be performed by any user, but only by personal with MDB CM access. A model developer does not have MDB CM access in general.
Therefore a CSS user should assure that following items are already created in the MDB before starting a model editing session.

Following MDB items are needed

* a CCU which references at least one CDU used for model development
* a CDU of domain CSS for the models

*Note that the MDB user who performs the model development must be the **owner** of the CDUs.*

This has to be checked first before a model editing session can be started successfully.

To connect the model to the 'outer world' at least one additional CDU is required. It must contain the stimuli (commands) to be sent to and the measurements to be received from the model.

For the first model development steps and model testing purposes this CDU is not mandatory.

7.4.4.1 Starting a Model Editing session

The fist action to do is to invoke the appropriate CGS tool supporting the mission configuration phase. This tool is known as I_MDB (Interactive Mission Database access) and will provide the user with a window called 'I_MDB Navigator'.

Starting a Mission Configuration Session

- **Move** the **mouse cursor** to the menu ***Select Task*** of the ***CGS Task Selector*** window.
- **Hold the right mouse bottom**. You get now a menu with all tasks you may select. (see Figure 7–37).
- **Move** the **mouse cursor** to the task ***Mission Preparation*** and **release** the **right mouse button**.

Note, the contents of the task list is not fixed and can be modified for each user by editing the '.task_list' file. Figure 7–37 shows the standard CGS Task Selector

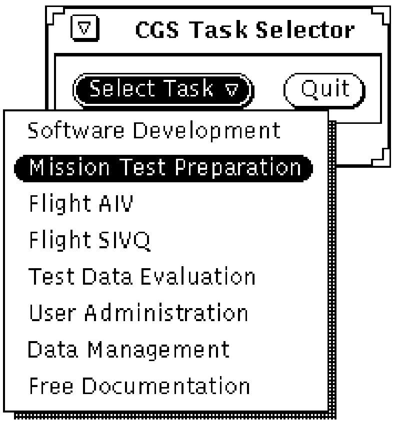


Figure 7–37 : *Mission Configuration Start from CGS Task Selector*

Select the System Tree version

Navigating down within a System Tree

Note, that the use of the Scroll-Bars may be necessary, but is not explicitly mentioned.

- **Double click** on the element you want to navigate to. This results to a box called ***System Tree Versions*** listing all available version of the element system tree and the corresponding CM status.
- **Double click** on the version you want to navigate to. Now the I_MDB Navigator window is updated as follows :
  - The Current-Path shows the Element Name and its system tree version.
  - The Node-List provides a list of all system tree nodes forming part of this version

Select the CCU

By selecting a CCU version a special view on the element configuration tree is created, providing only those CDUs contained in the selected CCU version. This makes it easier to find a specific CDU in the MDB.

A CCU can be selected either from a complete list of all CCU versions defined for the selected element or from a list of all CCU versions defined for a particular System Tree node :

Navigating down to a CCU version from the complete list

- **Move the mouse** to the Menu-Line and **select File–>Browse All CCU Versions...** . This opens the ***Browse CCU Versions*** box listing <u>all</u> defined CCUs including pathname, CCU Name and CCU version.
- **Use the Scroll-Bar** on the right site of ***Browse CCU Versions*** if necessary.
- **Click** on the CCU you want to access and **click** on the ***Apply*** button.

For more information about navigating to a CCU refer to chapter 6.1.2.3.

Select the CDU

For the procedure below it is assumed that a system tree version of an Element Configuration has be accessed and a CCU has been selected which contains the desired CDU.

Navigating down within a User Tree Version

Note, that the use of the Scroll-Bars may be necessary, but is not explicitly mentioned.

- **Double click** on the items in the tree structure until you reach the desired CDU.
- **Double click** on the CDU to be selected. This action leads to the I_MDB Navigator window which now lists :
  - The path to the CDU version as the Current-Path
  - A list of models forming part of the CDU version in the Node-List

Create a new model (Toplevel Function block)

In the beginning the selected CDU is empty i.e. there is no list of models. The users first action is to create a top–level function block (this is the name for the highest level of a model in the MDB terminology).

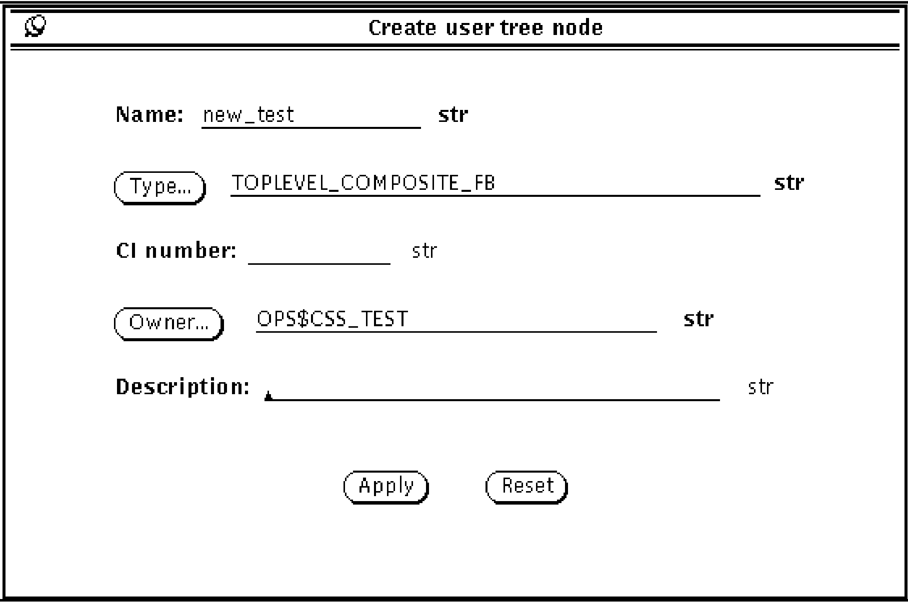


Figure 7–38 : *The Create user tree node window*

Creating a new model

- Press the **File** button with the left mouse button and select **Create Node...** from the pull–down menu. The ***Create user tree node*** window appears. (see Figure 7–38)
- Type the **model name** in the name field.
- Press the **Type** button. The ***Node Type list help*** window pops up. (see Figure 7–39)
- Select the type **TOPLEVEL_COMPOSITE_FB** and press the **Apply** button.
- Type a **model description** in the description field.
- Press the **Apply** button. The new model name appears in the I_MDB window.

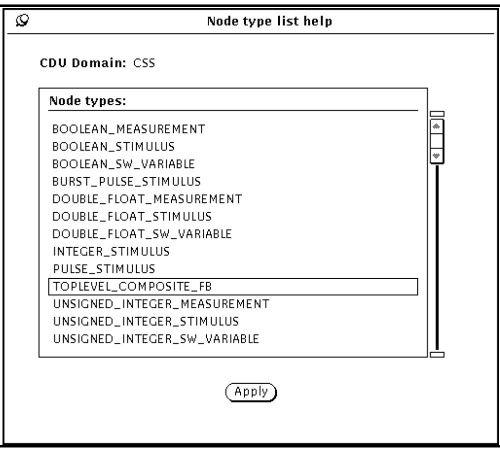


Figure 7–39 : *The Node type list help window*

7.4.5 Starting CSS User Interfaces

When invoked from I_MDB, CSS will automatically open the pre–selected simulation model for editing or execution. Additionally CSS presents itself with the data base browser (DBB) for browsing through CSS relevant data in the data base (f.e. CDUs containing CSS models), and finally routing to MDE or MOCS after having selected data for editing resp. execution and observation.

Summarizing CSS provides the following three user interfaces:

* DBB (Data Base Browser)
* MDE (Model Development Environment)
* MOCS (Model Observation and Control System)

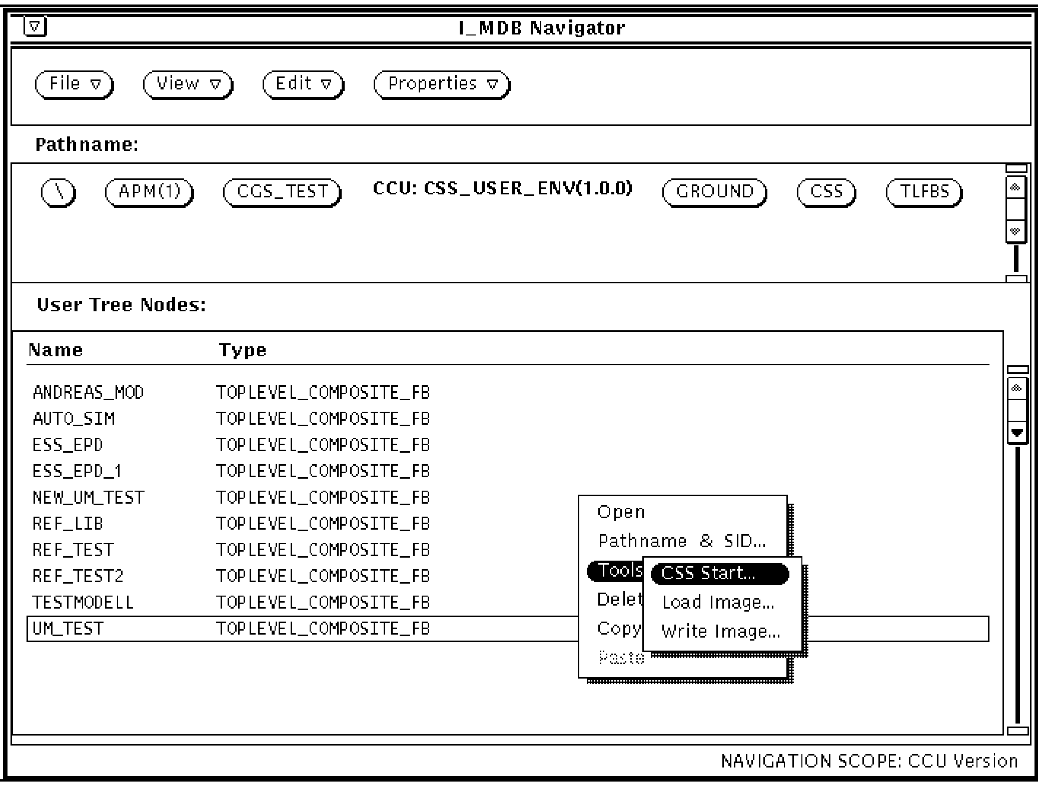


Figure 7–40 : *The CDU with a list of models*

Start the CSS User Interface for Model editing

Start the CSS User Interface for Model editing

■ **Select** the model in the I_MDB window. (see Figure 7–40)

■ Press the left mouse button and select **CSS start...** from the pop–up menu.

■ The CSS scope check window appears. MDE is pre–selected. Scroll to the bottom of the message area to see the latest appended message. If no unresolved references are reported you can press the **Start CSS...** button.

■ The Data Base Browser window and the Composite Editor window will be opened automatically.

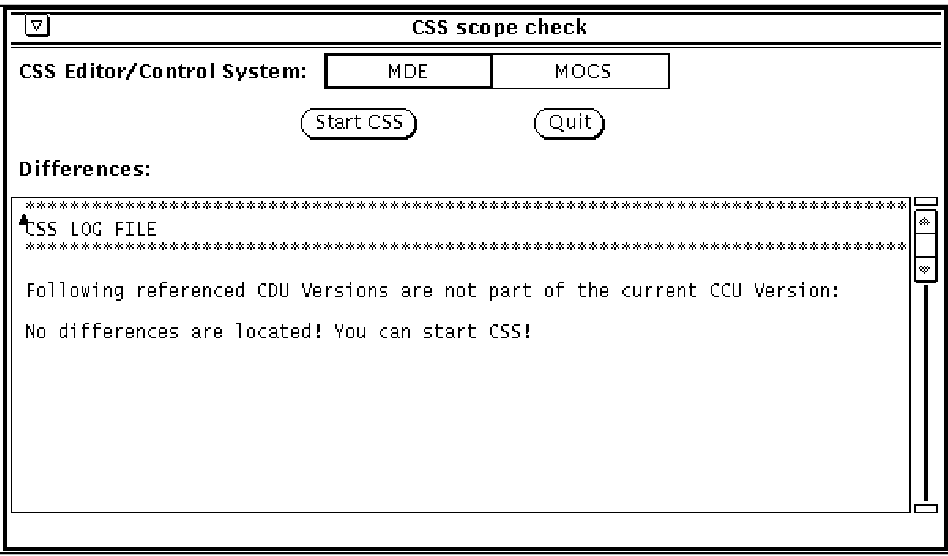


Figure 7–41 : *The CSS scope check window*

Start the CSS User Interface for Model editing with unresolved references

Start the CSS User Interface for Model editing

- **Select** the model in the I_MDB window. (see Figure 7–40)
- Press the left mouse button and select **CSS start...** from the pop–up menu.
- The CSS scope check window appears. MDE is pre–selected. Scroll to the bottom of the message area to see the latest appended message. The CSS scope check window reports the missing CDU versions (see Figure 7–42). Nevertheless you can press the **Start CSS...** button.
- A decision window pops up (see Figure 7–43). If you decide to open the model, press the **yes** button.

Note that updating the invalid function references may be a really time consuming procedure

- The Data Base Browser window and the Composite Editor window will be opened automatically.

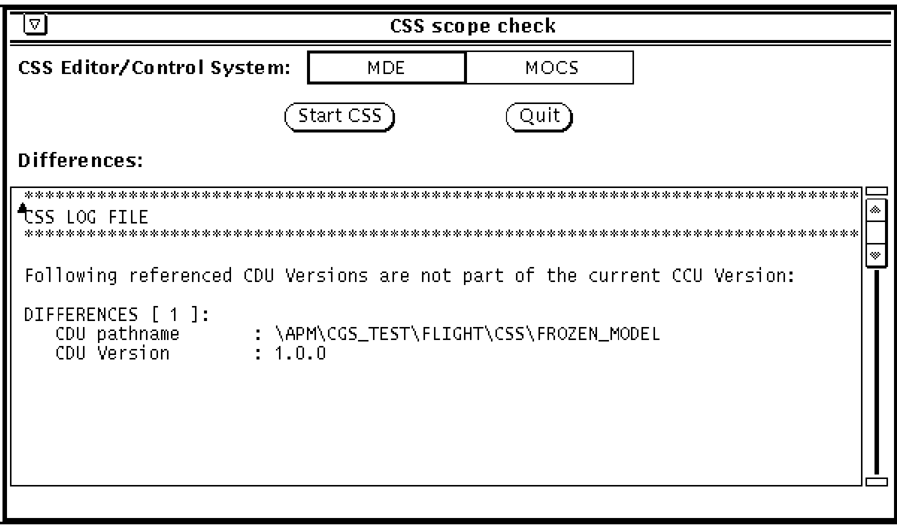


Figure 7–42 : *The CSS scope check window reports missing references*

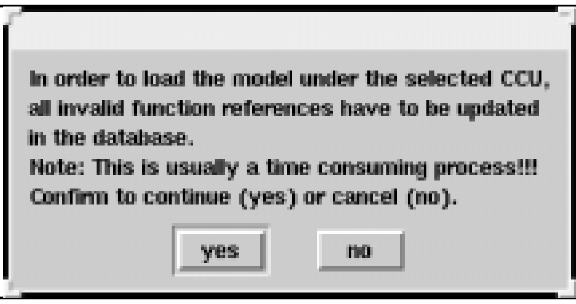


Figure 7–43 *: The CSS decision window*

7.4.5.1 Restrictions on model editing

The following functions are not yet available in the current implementation.

- The necessary functions to use CSS type ”COMPLEX” are not yet implemented.

7.4.6 Database Browser User Interface

7.4.6.1 General

The DBB is a subsystem of the CSS User Interfaces which allows to browse through the underlying database both for simulation models and for definitions of onboard items (i.e. stimuli and measurements).

For simulation models, CSS tools for inspecting and/or editing simulation models, configuring executable simulator kernels (MDE), inspecting and/or editing simulation tables, starting, controlling and monitoring simulations (MOCS) may be invoked.

Definitions of onboard items may be selected and exported to the relevant MDE tools (i.e. to create the references from simulation models to onboard items).

7.4.6.2 DBB Master Window

The DBB user interface consists of a single master window, tiled into several subviews.

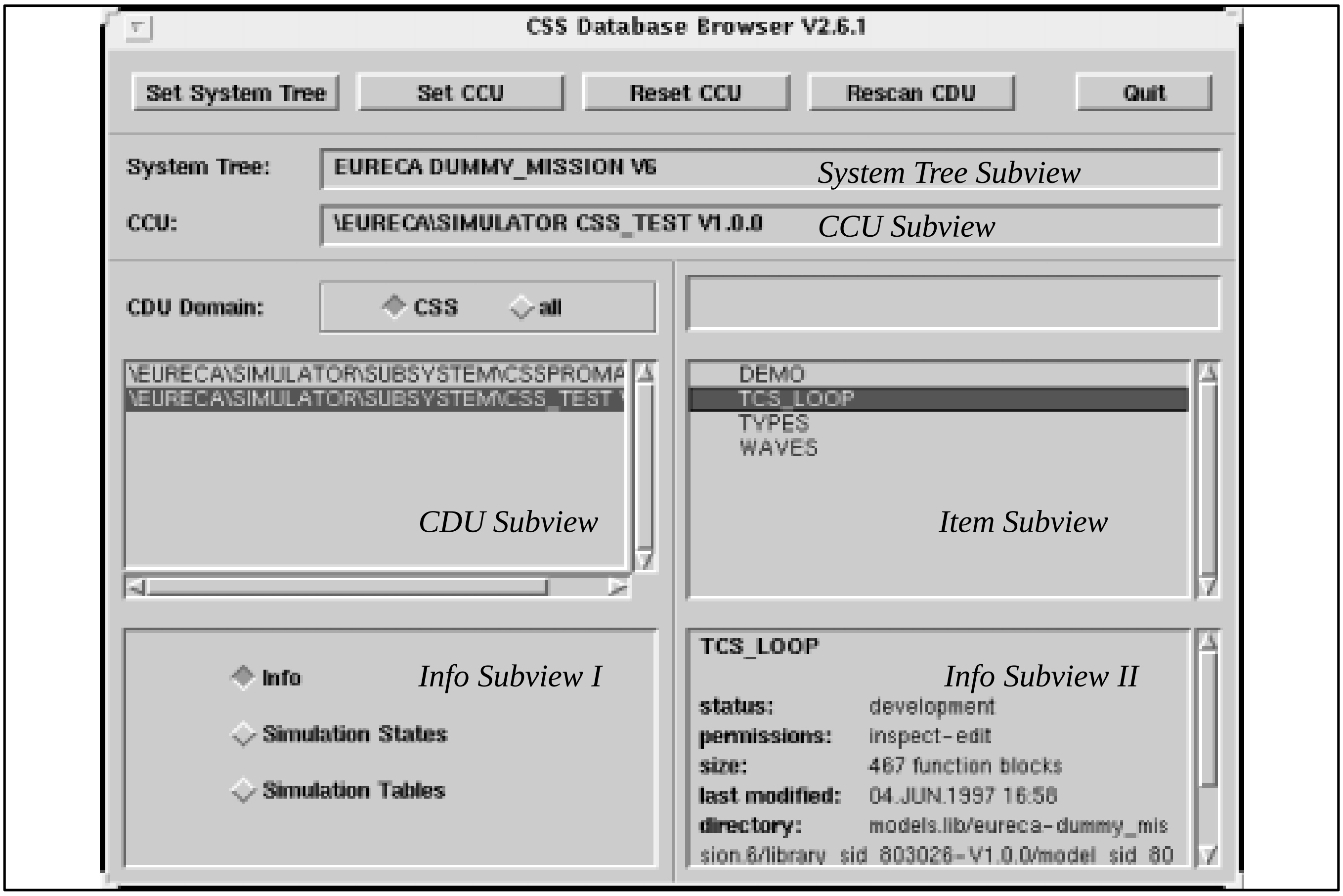


Figure 7–44 : *DBB Master Window with selected model*

The DBB forms the central part of the CSS User Interfaces (i.e., it controls the execution of all other tools); it is the only tool which is continuously present throughout the lifetime of the CSS User Interfaces operating system process (even if potentially in an iconized state). It is therefore the tool which provides the functionality to quit the CSS User Interfaces.

In the following, a description of the contents of the individual DBB subviews is presented:

Tool Bar

The **Set System Tree** button allows to select a system tree in the MDB. A selection window appears listing all available system trees.

The **Set CCU** button allows to select a CCU in the currently selected system tree. A selection window appears listing all available CCUs; it provides buttons allowing to restrict the CCUs listed to those that contain the CDU which is selected in the CDU subview, either in any or in the specific version.

If a CCU is selected, the CDU subview presents only the CDUs that are contained in the respective CCU.

Note that a CCU has to be selected in order to invoke the various CSS tools.

The **Reset CCU** button allows to deselect the CCU. If no CCU is selected, the CDU subview presents all CDUs in the selected system tree.

Note that as long as no CCU is selected, it is not possible to do any further work.

The **Rescan CDU** button allows to invalidate cached database information, causing the tool to re–initialize the display with updated values from the database.

The **QUIT** button quits the CSS session, ensuring that all tasks have been properly finished.

System Tree Subview

The system tree subview identifies the currently selected system tree. First part is the element configuration name (which is the first element in the system tree) (here ”**EURECA**”), followed by the mission name (here ”**DUMMY_MISSION**”) and the system tree version (here ”**V6**”, see Figure 7–44).

CCU Subview

The CCU subview identifies the currently selected CCU. First comes the pathname of the node in the system tree where the CCU is located (here ”**\EURECA\SIMULATOR**”), then the name of the CCU (here ”**CSS_TEST**”) and the CCU version (here ”**V1.0.0**”, see Figure 7–44).

CDU Subview

This subview shows the list of CDUs contained in the previously selected CCU. If no CCU is selected, all CDUs in the current system tree are listed.

In order to access simulation models, select the **Domain: CSS** button. The CDUs listed are filtered by domain CSS (the pop up menu of the item subview provides commands to create, delete and modify simulation models). Otherwise, to select an onboard item for referencing, select the **Domain: all** button. All CDUs regardless of their domain are listed (the pop up menu of the item subview provides commands for navigating in trees of virtual nodes).

Item Subview

The item subview shows the list of nodes under the currently selected CDU.

In a CDU of domain CSS, the nodes listed are simulation models. If the **Domain: CSS** button is selected, the pop up menu of this subview provides a variety of operations on simulation models depending on CDU resp. model status and ownership. In contrast to database based simulation models (i.e. simulation models the sources of which are kept in the database), filesystem based simulation models (i.e. simulation models the sources of which are kept in the filesystem) are marked with an icon. The sub-

view allows multiple selection: Point to an item and click the middle mouse button to select it in addition to previously selected items.

In a CDU of a domain other than CSS, the nodes listed may be of any type. For CSS, only virtual nodes and onboard items (stimuli and measurements) are of interest. If the **Domain: all** button is selected, the pop up menu of this subview provides commands to navigate in trees of virtual nodes.

Info Subviews I and II

The contents of the Info Subviews I and II depend on the selections in the other subviews. Per default (if a CDU, but no item is selected), Info Subview I shows information about the currently selected CDU, such as status (development/revision/frozen), domain and owner, Info Subview II is empty.

If the **Domain: CSS** button is selected and a model is selected in the Item Subview, Info Subview I shows a number of buttons allowing to select a category of model attributes:

- **Info**
- **Simulation States**
- **Simulation Tables**

If the button **Info** is selected, Subview II shows information about the selected simulation model, i.e. model status, the permissions, the architectures for which the model is configured, the model size and modification date.

If one of the buttons **Simulation States** or **Simulation Tables** is selected in Info Subview I, the list of simulation states or the list of simulation tables belonging to the selected model is displayed in Info Subview II. The pop–up menu in Info Subview II gives access to the creation, modification and deletion of simulation tables resp. the deletion of simulation states.

If the **Domain: all** button is selected and an item is selected in the Item Subview, Subview II shows information about the selected item.

7.4.6.3 Accessing Simulation Models

Simulation models may be database or filesystem based. The source of a database based simulation model is kept in the database. If such a model is loaded, its references to onboard items that have become undefined or incompatible are detected. If appropriate, undefined onboard references may be adapted interactively, see section 7.4.7.8, otherwise they will be reset automatically. However, if the model is relatively large, the process of loading may be time consuming.

The source of a filesystem based simulation model is kept in the filesystem. The process of loading such a model is usually less time consuming than that of a database based model, specifically for large models there is a significant performance advantage. However, function block references don't work with these models, also references to onboard items that have become incompatible to the referencing interface items are not detected and reset when the model is loaded.

Database based simulation models may be exported into the file system and vice versa, filesystem based simulation models may be imported into the database.

Names of models, simulation tables and simulation states must comply with Ada identifier syntax, i.e.:

* the first character must be a letter
* any subsequent characters must be letters, digits, or the underscore ('_')
* two underscores cannot occur together, nor can a model name end with an underscore

Names of simulation models must be unique in the scope of a CDU, with the exception that a database based and a filesystem based simulation model may have the same name. Names of simulation tables must be unique in the context of a specific simulation model, the same is true for simulation states.

Creating a simulation model

- **Select** the **Domain: CSS** button.
- In the Item Subview, deselect and choose **add model–>in database** or **add model–>in filesystem** from the pop–up menu, depending on whether you want to create a database based or a filesystem based simulation model.
- An input window appears: Type in the name of the new simulation model.
- The new simulation model appears in the Item Subview and is automatically selected.

Copying a simulation model

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to copy and choose **copy** from the pop–up menu.
- Switch to another CDU, possibly in another system tree and/or in another CCU.
- In the Item Subview, deselect and choose **paste** from the pop–up menu.
- If a naming conflict occurs, an input window appears: Type in the name of the copied simulation model.
- The copied simulation model appears in the Item Subview and is automatically selected.

Renaming a simulation model

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to rename and choose **rename** from the pop–up menu.
- An input window appears: Type in the new name of the simulation model.
- The simulation model’s new name appears in the Item Subview, the model stays selected.

Deleting simulation models

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model(s) you want to delete and choose **delete** from the pop–up menu.
- The selected simulation model(s) disappear(s) from the Item Subview.

Exporting a simulation model from the database into the file system

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the database based simulation model you want to export and choose **export** from the pop–up menu.

- If a naming conflict with another filesystem based simulation model occurs, an input window appears: Type in the name of the exported filesystem based simulation model.
- The new filesystem based simulation model appears in the Item Subview and is automatically selected.

Importing a simulation model from the filesystem into the database

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the filesystem based simulation model you want to import and choose **import** from the pop–up menu.
- If a naming conflict with another database based simulation model occurs, an input window appears: Type in the name of the imported database based simulation model.
- The new database based simulation model appears in the Item Subview and is automatically selected.

Not all parts of a configured database based simulation model are stored in the database. The model's executable images (simulator kernels), the simulation states, the mapping tables and the adaptation system configuration files for the various architectures are kept under the user's own account in the UNIX filesystem.

If simulation models shall be part of a data contents transfer between MDB instances via the MDA import/export mechanism, they must be stored in the database completely. The **pack** command allows to store the data normally kept in the filesystem into the database. The corresponding files are deleted automatically. A packed simulation model is ready for an export.

Packing configured database based simulation models in the database

- **Select** the **Domain: CSS** button.
- In the Item Subview, **select** the model(s) and choose **pack** from the pop up menu.

It is not possible to do any further work with a packed simulation model unless it is unpacked; this has to be done after an import has been taken place. The **unpack** command allows to revert the effect of the pack operation, i.e. to reestablish the corresponding data in the filesystem.

Unpacking configured database based simulation models in the database

- **Select** the **Domain: CSS** button.
- In the Item Subview, **select** the model(s) and choose **unpack** from the pop up menu.

Note that the pack and unpack operations always comprise all architectures the model has been configured for.

CSS provides a number of tools that can be opened on a simulation model from within the Database Browser: the Composite Interface Inspector, the Composite Inspector, the Composite Interface Editor, the Composite Editor, the Hierarchy Browser, the Documentation Tool and the Simulation Controller.

Opening a tool on a simulation model

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to open a tool on and choose one of the commands **inspect–>interface**, **inspect–>implementation**, **edit–>interface**, **edit–>implementation**, **browse hierarchy**, **print** or **execute** from the pop–up menu.

- The tool opens in a separate window.

Simulation tables and simulation states are attributes to simulation models.

Creating a simulation table

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to add a simulation table to.
- In the Info Subview I, select the **Simulation Tables** button.
- In the Info Subview II, deselect and choose **add table** from the pop up menu.
- An input window appears: Type in the name of the new simulation table.
- The new simulation table appears in the Info Subview II and is automatically selected.

Copying a simulation table

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to copy a simulation table from.
- In the Info Subview I, select the **Simulation Tables** button.
- In the Info Subview II, select the simulation table you want to copy and choose **copy** from the pop up menu.
- Select the simulation model you want to add the copied table to, possibly in another system tree and/or in another CCU and/or in another CDU.
- In the Info Subview II, deselect and choose **paste** from the pop–up menu.
- If a naming conflict occurs, an input window appears: Type in the name of the copied simulation table.
- The copied simulation table appears in the Info Subview II and is automatically selected.

Renaming a simulation table

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to rename a simulation table from.
- In the Info Subview I, select the **Simulation Tables** button.
- In the Info Subview II, select the simulation table you want to rename and choose **rename** from the pop up menu.
- An input window appears: Type in the new name of the simulation table.
- The simulation table’s new name appears in the Info Subview II, the table stays selected.

Deleting a simulation table

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to delete a simulation table from.
- In the Info Subview I, select the **Simulation Tables** button.

- In the Info Subview II, select the simulation table you want to delete and choose **delete** from the pop up menu.
- The selected simulation table disappears from the Info Subview II.

CSS provides two tools that can be opened on a simulation table from within the Database Browser: the Simulation Table Inspector and the Simulation Table Editor.

Opening a tool on a simulation table

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to open a tool on a simulation table from.
- In the Info Subview I, select the **Simulation Tables** button.
- In the Info Subview II, select the simulation table you want to open a tool on and choose **inspect** or **edit** from the pop up menu.
- The tool opens in a separate window.

Deleting a simulation state

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to delete a simulation state from.
- In the Info Subview I, select the **Simulation States** button.
- In the Info Subview II, select the simulation state you want to delete and choose **delete** from the pop up menu.
- The selected simulation state disappears from the Info Subview II.

7.4.6.4 Selecting an Onboard Item

In contrast to simulation models, onboard items (i.e. stimuli and measurements) are usually not located as subnodes directly under a CDU. Instead, possibly together with end items of various types, they may be organized in a tree structure of virtual nodes. To select an onboard item, it is necessary to navigate in this tree.

Make sure that a CCU is selected and press the **Domain: all** button in the DBB window. The CDU Subview shows the list of all CDUs contained in the current CCU. Select one of the CDUs. The list of subnodes is displayed in the Item Subview. If an item is selected, Info Subview II shows information about the selected item, at least whether it is a virtual node, a simulation model or an unknown end item. If the item could be identified as an onboard item, it is stated whether the item is a stimulus or a measurement and further information is provided, i.e. the corresponding CSS data type and state codes or engineering unit, if appropriate.

If a virtual node is selected, a pop up menu command **expand** allows to view the subnodes unter that virtual node. The pathname (relative to the selected CDU) of the current virtual node is presented on top of the Item Subview; the Item Subview provides the subnodes under that virtual node. If no item is selected in the Item Subview, the pop up menu command **collapse** allows to step back to the next higher level in the tree of virtual nodes.

7.4.7 MDE User Interface

7.4.7.1 Composite Editor

7.4.7.1.1 Basics

The Composite Editor is activated in a separate MDE window. It is the basic window for an edit session on a particular simulation model. Atomic Editors and Icon Editors can be started from within the Composite Editor. Only one Composite Editor may be opened on a specific model.

User Interface

Figure Figure 7–45 shows the user interface of the composite editor. The tool provides a composite edit view (*edit view*) that allows interactive model editing using MDE–GL. It provides vertical and horizontal scrolling functions. Additionally to the normal edit mode the view can be run in overview mode. A bar of iconized buttons allows the user to influence the behaviour of the composite edit view (e.g. to specify the MDE–GL syntax element that should be created next, to switch from edit mode to overview mode and vice versa). A read only text view is used to provide textual user feedback (*message subview*).

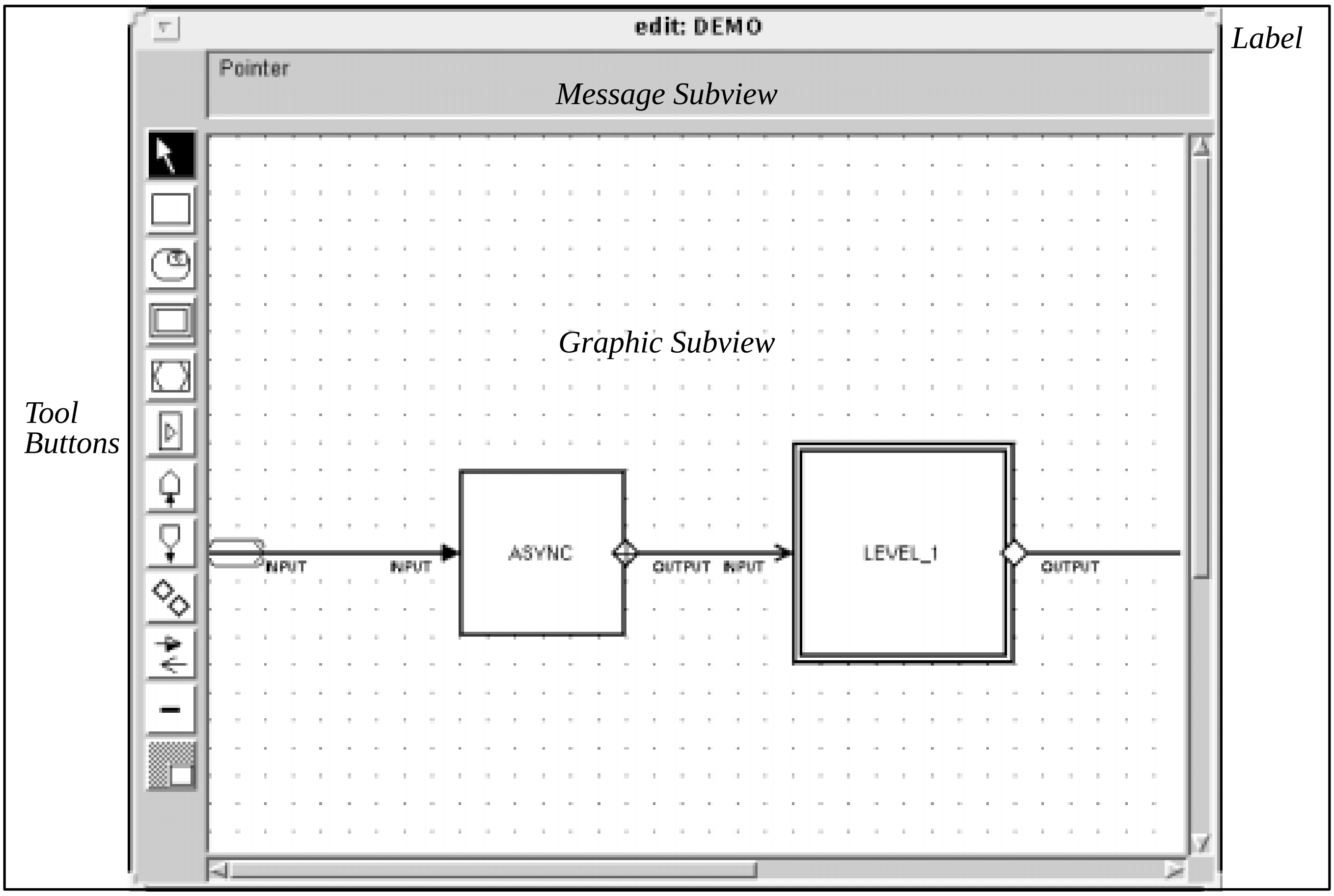


Figure 7–45 : *The Composite Editor User Interface*

7.4.7.1.2 The Composite Editor’s Components

7.4.7.1.2.1 The Label

The label of the Composite Editor shows the pathname of the composite function block that’s inside view is currently displayed in the graphic subview. It is called the current level in the hierarchy of composite function blocks.

When you open a Composite Editor the model itself is initially set to the current level.

7.4.7.1.2.2 The Graphic Subview

The scrollable graphic subview shows a part of the inside view of the composite function block that’s pathname is shown in the label. Note that the inside view of the just created composite function block is empty.

Move the cursor into the scroll bar area. Press the left button and hold. The bar moves to the cursor location and then tracks the cursor until the left button is released. The displayed document jumps to the appropriate location.

The scrollbars provide additional buttons. Clicking on them allows scrolling step by step in the related direction.

7.4.7.1.2.3 The Tool Buttons

The operation initiated by pressing the middle mouse button inside the graphic subview depends on which of the tool buttons is currently activated. The active tool button appears highlighted. Select different buttons by moving the cursor over the button and clicking the left mouse button. Note that only one tool button can be active at a time, i.e. the activation of a particular button automatically deactivates the previously active button.

Note that clicking the left button inside the graphic subview always sets the pointer button to be the active button.

Editorial Note:
Since the Frame Synchronization Point has been added only recently, most screen snapshots of the model editor window in this chapter have not been updated to include the corresponding icon.

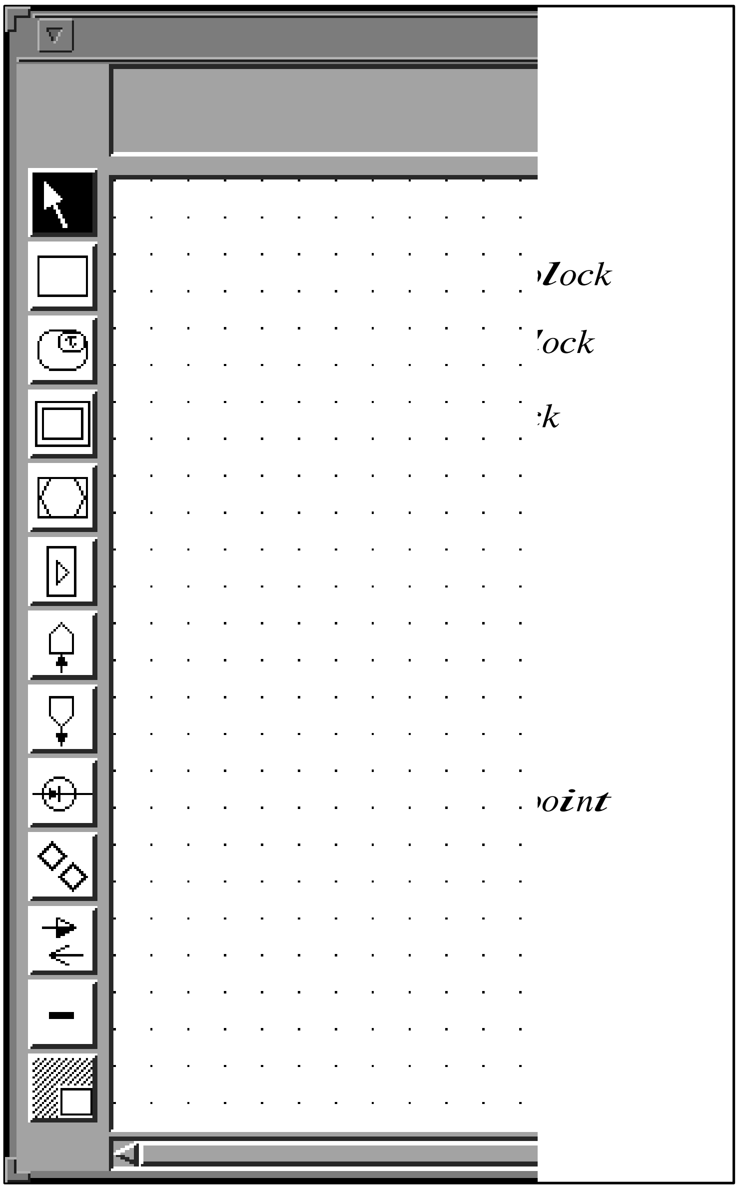


Figure 7–46 : *The tool buttons of the Composite Editor*

7.4.7.1.2.4 The Message Subview

While editing the message subwindow is used to provide feedback. For example if you try to perform an illegal action you will be informed by a message indicating the mistake.

7.4.7.1.2.5 The Overview Mode

In general the editor window will be not large enough to display the entire graphical document (i.e. the inside view of the actual displayed composite function block).

The *overview mode* allows you to see the whole document at a time. It also provides a convenient way to scroll to a particular location. To enable overview mode, activate the overview button.
The actual visible part of the document is indicated by the white underlaid rectangle while the other parts are grey underlaid.

You can change the visible part by scrolling horizontal or vertical using the particular scroll bar.

A more convenient way is to move the cursor inside the white underlaid area and press the left button. Note that the cursor shape changes to that of a hand. Holding the button pressed move the cursor around.

To go back to *edit mode* press one of the tool buttons (except the overview button) or select **edit** from the pop–up menu also available by pressing the right mouse button.

The previously white underlaid part of the document becomes the visible part in the graphic subview.

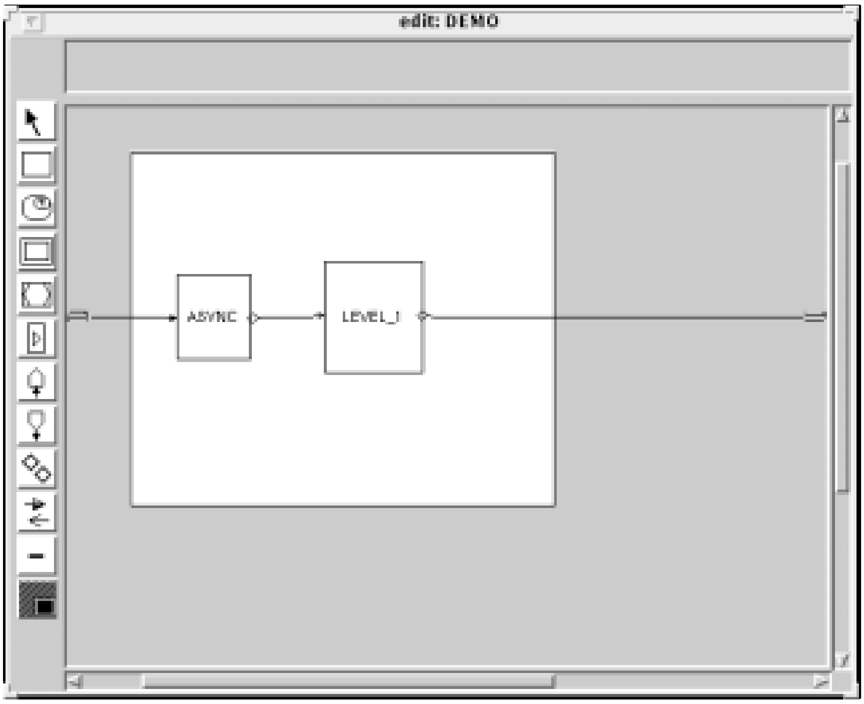


Figure 7–47 : *Scrolling in overview mode*

7.4.7.1.3 Selection Sensitive Menus

There are a number of different pop–up menus. The menu that actually appears when you press the right mouse button depends on whether or not there are selected objects and what types of objects are selected. Even the numbers of selected objects can affect the appropriate menu.

7.4.7.1.4 The Composite Editor’s Basic Menu

Move the cursor inside the graphic subview and press the right mouse button. You obtain the Composite Editor’s basic menu (i.e the menu that appears when there are no selected objects).

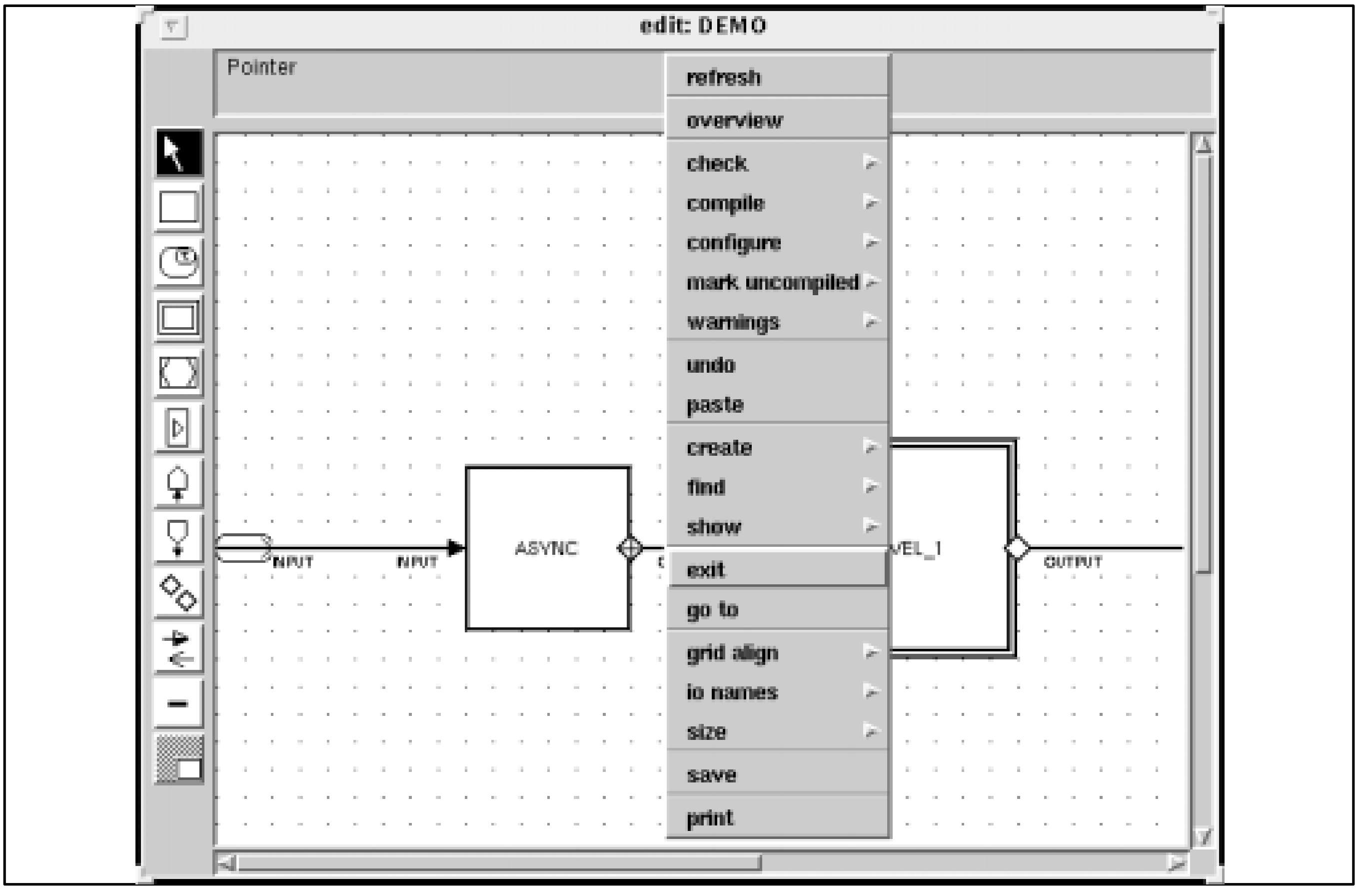


Figure 7–48 : *The Composite Editors basic menu*

7.4.7.1.5 Creating Block Objects, Grouping Entries, Global Symbols and Frame Synchronization Points

Remember that the actions invoked by pressing the middle mouse button depend on which tool button is activated. Activate the tool button labeled with an asynchronous function block by moving the cursor over the tool button and clicking the left mouse button. Now you can create one or more asynchronous function blocks.

To do so, move the cursor inside the graphic subview and click the middle button. A black rectangular outline appears on the screen. The size of this rectangle indicates the minimum size of the specific kind of object, in this case the minimum size of the graphical symbol for an asynchronous function block. Move the cursor around without pressing a button.

Note that the black rectangular outline tracks the cursor. Move it to the location you want to draw the asynchronous function block and press the middle button. If you hold the middle button and move the cursor you can resize the just created function block. Release the mouse button, an asynchronous function block appears on the screen.

The last created object is automatically selected. In this case this is indicated by black selection marks. Create another asynchronous function block. See that the first function block is automatically deselected.

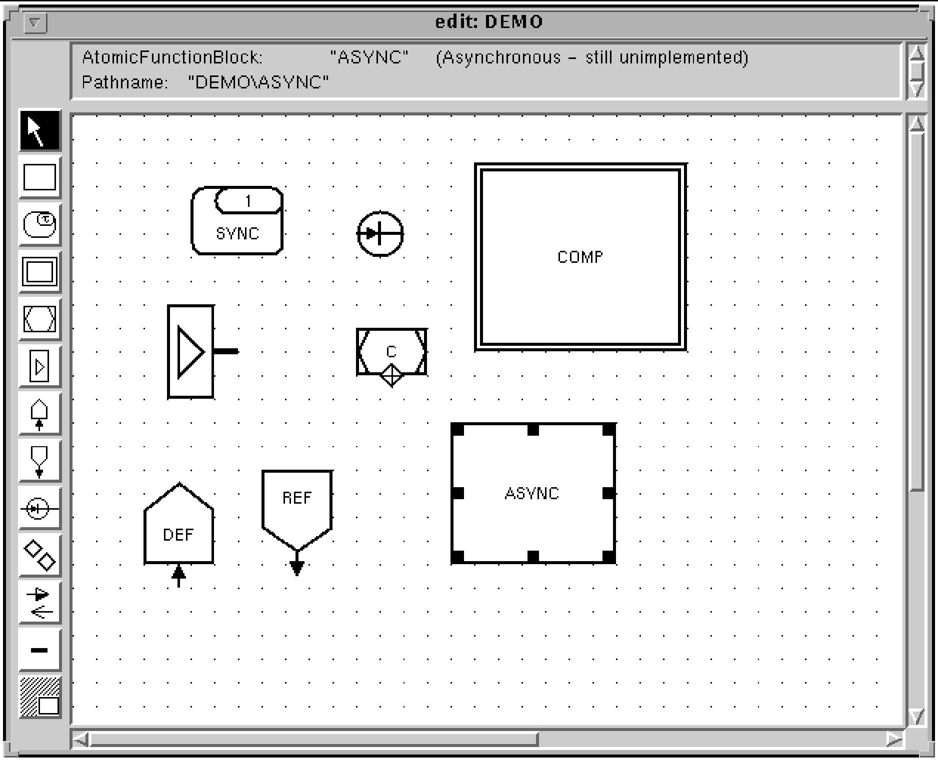


Figure 7–49 : *A Composite Editor showing some block objects. The last created asynchronous function block is selected.*

The same way you can create synchronous and composite function blocks as well as constant blocks and grouping entries. For each of these objects there is an appropriate tool button. Note that each constant block is created including an output symbol.

Definition point, reference point and frame synchronization point are created in a similar way. The only difference is that you can’t specify a size by moving the pressed middle button. Click the middle button to get the black rectangular outline. Move it to the desired location and simply click the middle button again.

Another way to initiate creation of one of these objects is to choose the basic menu command **create**. To do so, move the cursor inside the graphic subview to an empty location and click the left mouse button to deselect all objects. Then press the left mouse button to get the basic menu. Drag the cursor up to **create**. A submenu appears listing the different objects you can create.

Choose the submenu item **composite** to create a composite function block. The black frame appears on the screen inviting you to determine location and size of the graphical symbol. Simultaneously the tool button labeled with a composite function block is set to be active.

7.4.7.1.6 Creating Outputs, Inputs, Grouping Links

To create an output, input or a grouping link, activate the appropriate tool button by moving the cursor over the button and clicking the left mouse button.

Each i/o–item you create has to be attached to an asynchronous, synchronous, composite function block or a grouping entry (grouping links can only become attached to composite function blocks).

You specify the target function block by moving the cursor over its symbol. Then click the middle button. A black frame appears that can be moved on the function block’s border lines by moving the cursor around.

Move the frame to a place you want the i/o–item to appear. Click the middle button once again to finish creation.

If the cursor is not positioned over a particular function block, then the function block nearest to the cursor position is assumed to be the destination function block.

The graphic subview always represents the inside view of the composite function block you are currently editing. You can also assign i/o–items to this composite. Move the cursor near the border of the inside view but don't leave the graphic subview (maybe you first need to scroll to reach the border of the inside view). This has the effect that the currently edited composite is supposed to be the target function block. Click the middle button. Move the appearing black frame to a suitable position and click the middle button again.

Note that the last created i/o–item is automatically the selected object. The selection of an input or grouping link is indicated by a black dot at the point of the arrow or line.

Assign an input to an atomic function block.

- *Note that its default activation mechanism is always.*

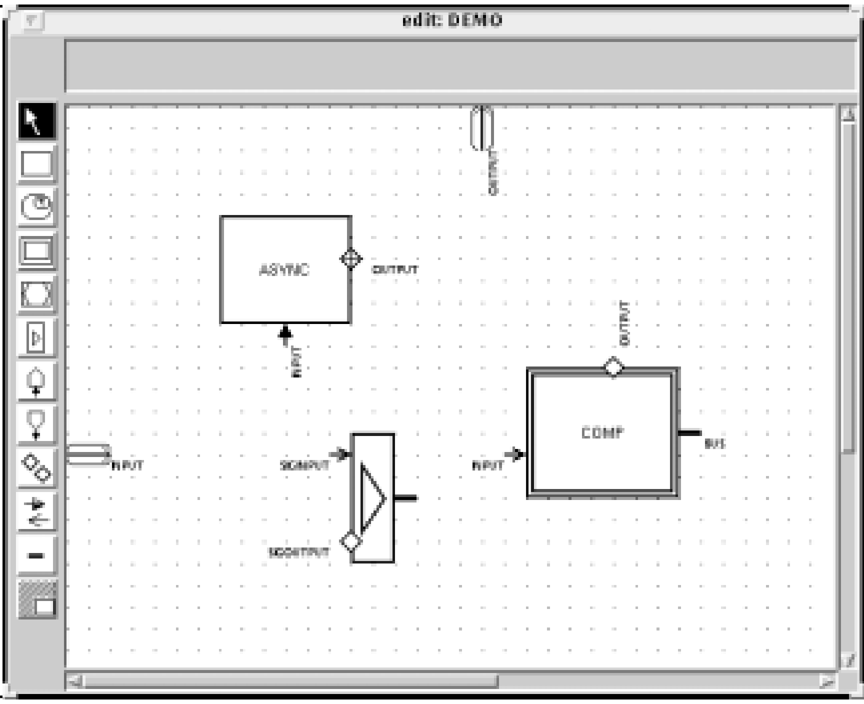


Figure 7–50 : *A Composite Editor showing some I/O items attached to block objects. The last created output is selected.*

Create a number of i/o–items to a function block. Note that you can't specify a location that another i/o–item occupies. Note also that the maximum number of i/o–items depends on the size of the function block's graphical symbol. If this limit is reached and you want to create a further i/o–item, the operation will become aborted. You will be informed by a message in the message subwindow.

7.4.7.1.7 Selecting Objects

There are two different ways of selection. Exclusive selection means that the object you want to select should be the only selected object (i.e. eventually previously selected objects should automatically become deselected). Additional selection means that an object should become selected in addition to other selected objects.

Exclusive selection is performed by moving the cursor over the graphical symbol and clicking the left button; additional selection is performed similarly by use of the middle button.

Note that additional selection is restricted to objects of the same type. So you can not select an output in addition to function blocks. If you try, the output will become selected while the function blocks are automatically deselected.

The selection of a function block involves the selection of the related i/o – items, connection lines and i/o – item names (not visibly selected) in the context of moving and deleting function blocks.

7.4.7.1.8 Deselecting Objects

To deselect all selected objects at once simply move the cursor inside the graphic subview to an empty location and click the left button.

To deselect a specific object make sure that the pointer tool button is activated. Move the cursor over the selected graphic symbol and click the middle button.

7.4.7.1.9 Renaming Objects

Select the object you want to rename and press the right mouse button. Execute **rename**. A text field appears asking you for the new name. Type in the new name.

Note that there is no dedicated name for a frame synchronization point.

7.4.7.1.10 Resizing Block Objects

Select the object you want to resize. Move the cursor over one of the black selection marks. Press the left button. The cursor changes shape and a rectangular outline appears on the screen. Move the cursor around. Correspondingly the rectangle changes its size indicating a possible size for the graphical symbol. If you are satisfied, release the button. The resized block appears on the screen.

To change height and width of symbol at once, choose a selection mark that lies at one of the corners. The cursor changes shape to look like the appropriate corner of a rectangle.

This operation always maintains the minimum size of the function block.

Note that the minimum size of a function block depends on the number of i/o–items.

All block objects except definition point, reference point and frame synchronization point can be resized.

7.4.7.1.11 Rotating Block Objects

All block objects except definition point and reference point can be rotated. Select the object you want to rotate and execute the menu command **rotate**. Note that the function block rotates by 90 degrees counterclockwise.

7.4.7.1.12 Moving Block Objects

You can move a block object to another place inside the graphic subview. To do this you have to select the object first. Then move the cursor over the block (but not over a selection mark) and press the left button. A black frame indicating the borders of the block appears on the screen.

Move the cursor around holding the button down. See that the black frame tracks the cursor. Release the button if you are satisfied with the new position. The block appears at the new position.

You can also move several blocks at once. Select a number of blocks and position the cursor over one of these (but not over a selection mark). Press the left button and move the cursor around. For all objects one large frame tracks the cursor indicating its borders. Finish moving by releasing the left button. All selected blocks appear at their new position.

7.4.7.1.13 Moving I/O–Items

You can move a specific i/o–item to another place on the border lines of its associated function block. To do this you have to select the object first. Then move the cursor over the i/o–item and press the left button. A frame indicating the position of the i/o–item appears on the screen.

Holding the button pressed, move the cursor around. See that the frame tracks the cursor as long as a position is not occupied by another i/o–item. Release the button if you are satisfied with the new position. The i/o–item appears at the new position.

7.4.7.1.14 Placing Block Objects into the foreground resp. background

Block objects are allowed to overlap.

Create a number of these objects and move them so that they overlap partially. Select an object that is partially hidden. Choose the menu item **arrange**. A submenu appears listing two commands: **to front** and **to back**. Execute **to front**. Note that the selected block is placed into the foreground. Now execute **to back**. The selected object is placed into the background.

These operations can also be applied to a group of block objects.

7.4.7.1.15 Removing Block Objects

Select the object(s) you want to remove. Execute the menu command **cut**. The objects disappear.

The basic menu commands **undo** allows you to get back the last removed blocks.

7.4.7.1.16 Removing I/O–Items

Select the i/o–items(s) you want to remove. These may belong to different function blocks. Execute the menu command **cut**. The i/o–items disappear.

The basic menu command **undo** allows you to get back the last removed i/o–items.

7.4.7.1.17 Copying Block Objects

You can copy one object as well as several objects at once. Select the object(s) you want to copy. Execute the menu command **copy–>normal**. Then execute **paste**. Move the cursor around. Correspondingly the black rectangle (containing one or more copied objects) changes its position inside the graphic subview. Move the objects to a position where they don’t overlap other objects and click the middle button. The copies appear on the screen.

7.4.7.1.18 Copying Block Objects per reference

You can copy one object as well as several objects per reference from models which are in a CDU configured in your currently used CCU.

Copying block objects per reference

- The model editor window is open.
- Move the mouse cursor into the DBB window and **select** the CDU containing the model you want to reference in the CDU subview. The list of models appears in the models subview.
- **Click** on the desired model.
- Press the right mouse button and select **inspect –> implementation** from the pop–up menu. The inspector window is opened and shows the model in overview mode.
- Press the right mouse button and select **inspect** from the pop–up menu.
- Navigate to the desired part of the model.
- Select one or more block objects.
- Press the right mouse button and select **copy –> reference** from the pop–up menu.
- Move the cursor back to the editor window, press the right mouse button and select **paste**. A rectangle appears.
- Move the rectangle to the desired position in the model, then press the middle mouse button. The copied block objects appear with a grey underlay.
- Move the cursor back to the inspector window and select **Quit** from the window menu.
- Select the initial CDU in the CDU subview.

It is possible to set references to models which have the DB status develoment. Note the following restriction:

- as soon as the referenced model is stored (not necessarily changed !) all references are invalid i.e. next time the model is opened all (now invalid) references will be resolved, which can be a very time consuming procedure.

7.4.7.1.19 Changing the Activation Characteristics of Inputs

When you assign a new input to an atomic function block, its default activation characteristic is *always*. To specify another activation characteristic, select the input(s) and choose the menu item **activation**. A submenu appears listing the three different activation characteristics **always**, **on change** and **never**. Choose the desired characteristic. The graphical representation of the selected inputs will become updated.

*Note that the activation mechanism n **never** should only be used to connect constant blocks to asynchronous function blocks or as a timing control mechanism to avoid asynchronous loops.*
*Changing values transmitted via a connection with **never** characteristics will be automatically delayed for one time frame to avoid asynchronous loops.*

7.4.7.1.20 Stepping Through Composite Hierarchies

7.4.7.1.20.1 Stepping Into a Lower Level Composite Function Block

Select one composite function block and execute the menu command **enter**. The graphic area now shows the inside view of the particular composite function block. Note that the editor initially changes to overview mode. Note also that the label now shows the full pathname of the composite function block you stepped in. Go back to edit mode by choosing the menu command **edit** or by clicking the left mouse button over one of the tool buttons.

7.4.7.1.20.2 Stepping Back to the Next Higher Level

To go back to the parent composite function block, perform the following actions:
Deselect all objects to get the basic menu. Drag the cursor to **exit** and release the left mouse button. Note that the label changes the displayed name to that of the parent composite function block. Simultaneously the graphic subview displays the contents of the parent composite function block.

Note that there is one composite function block selected. This is the composite you just left.

7.4.7.1.20.3 The Tree Browser

Execute the basic menu command **go to**. A Tree Browser running in a separate window appears on the screen. It shows the composite hierarchy of the edited model.

Note : Since the collection of the data required for the Tree Browser display, especially for a large model, may take quite some time, the user is prompted for confirmation before the Tree Browser is actually started.

The Tree Browser allows to step to an arbitrary composite function block. Inside the Tree Browser, click the left mouse button on the name of the function block you want to go to. In the Composite Editor, execute **go to** once again. Correspondingly the Composite Editor changes the actual edited composite function block and selects the function block that is selected in the Tree Browser.

For more details about the Tree Browser user interface see the related chapter.

7.4.7.1.21 Connecting Objects

Select the objects you want to connect and press the right mouse button. If the selected objects are allowed to become connected, the menu provides a command **connect**. Execute **connect**. If ok, the items are connected graphically.

Note that the MDE prevents you from connecting type incompatible objects.

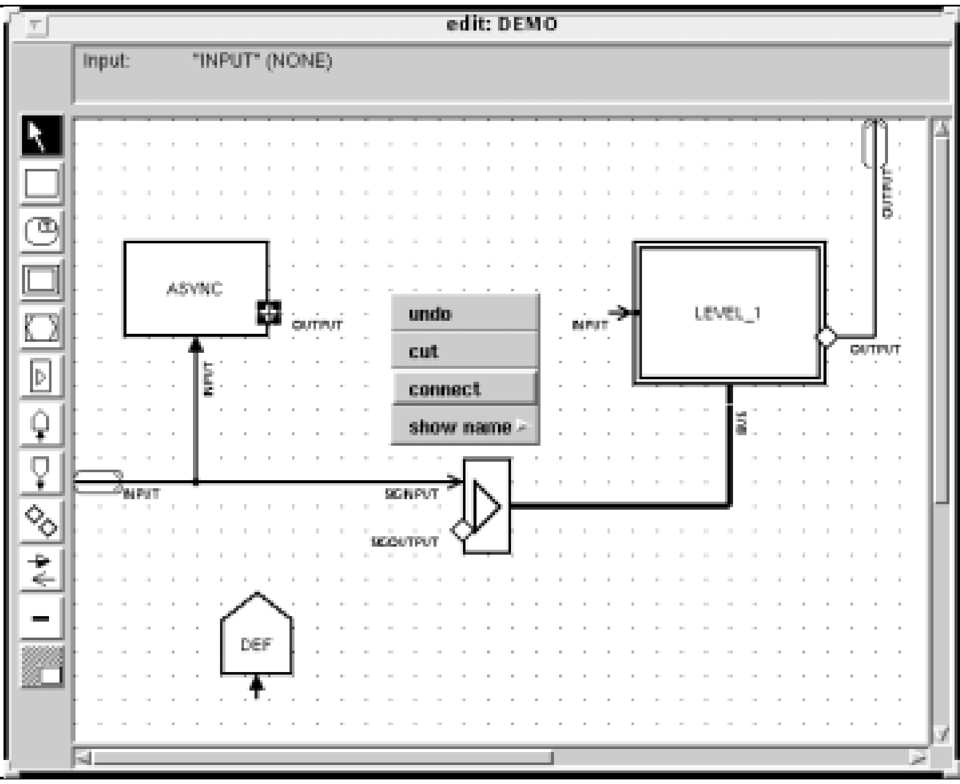


Figure 7–51 : *The output of function block ASYNC will be connected with the INPUT of COMP.*

7.4.7.1.22 Connecting top level I/Os to onboard items

If the model is intended to be used for H/W in the loop simulations, the connection between the simulator and the external H/W has to be defined by referencing onboard items specified in the database from the model's interface items (top level I/Os).
Following precondition has to be fulfilled: at least one CDU containing onboard items is configured in the CCU you selected previously.

☞ *The user needs at least two CDUs to get access to the model and to the onboard items.*

☞ *We recommend to create the onboard items in the database first and then to connect the onboard items with the CSS top level I/Os.*

If possible, MDE will check the compatibility of a model interface item and the referenced onboard item:

- When sufficient information is available from the MDB, and MDE finds both items are compatible, the top level I/O's symbol will be drawn in **green** color (as 'compatible reference').

  ☞ *The software type of an I/O item with type 'NONE' will be automatically adapted.*

- When MDE finds both items incompatible, or no sufficient information is available fom the MDB (e.g. for project specific end item types), the user will be asked to confirm the (possibly incompatible) reference. When confirmed, the top level I/O's symbol will be drawn in **red** color (as 'possibly incompatible reference').

  ☞ *Due to limitations in the CSS MDB interface, stimulus end items cannot be automatically checked and will therefore always be marked red (as 'possibly incompatible')*

- *For project specific end items, the corresponding adaptation system confuguration software is responsible for reporting incompatibilities between model top level I/Os and the referenced onbord items.*

- In case the user does not confirm the connection, no onboard reference is created, and the top level I/O’s symbol will remain drawn in **black** color (as ’no reference’).

Connecting top/level I/Os to onboard items

- The CSS model editor window is open.
- Top level inputs/outputs are created.
- *Note that top level I/Os can be undefined (no type setting, no initial value)*
- Move the mouse pointer into the DBB window and press the **all** button. (see Figure 7–52) A list of CDUs appears in the CDU subview.
- Select the desired CDU in the CDU subview (on the left side) by clicking on the name with the left mouse button.
- Navigate to the desired end item by selecting a virtual node in the right subwindow and then pressing the right mouse button and execute **expand**.
- *To navigate from an end item to a higher node, deselect the item, then press the right mouse button and select **collapse**.*
- Move the mouse pointer back to the model editor window.
- Select the I/O item by clicking on it with the left mouse button, then press the right mouse button and select **variable –> set onboard reference** from the pop–up menu.
- For top level inputs only: Press the right mouse button and select **variable –> initial value ...** from the pop–up menu. Type/select the initial value (a default initial value is set automatically).
- *Repeat the last four steps until all top level I/Os are connected..*
- Deselect the I/O then press the right mouse button and select **save** from the pop–up menu.

Figure 7–52 *: The stimulus LONG_REAL is selected in the database*

It is also possible to set onboard references from a file in ASCII format. The command **O/B References–>Set From File** is available if the Composite Editor is located on top level (i.e. it shows the MDE–GL implementation of the top level composite function block) and no item is selected. It allows to read such a file and to set the references of model interface items to onboard items as specified there.

The required format of such an ASCII file is described here:

- lines starting with # are considered as comments and are thus ignored
- each entry is stored in a separate line
- an entry consists of the following elements, all separated by one or multiple <space> and/or <tab>, respectively
  - model interface item name
  - the pathname of the CDU containing the referenced onboard item
  - the pathname of the referenced onboard item relative to the CDU
  - an optional comment, i.e. some explaining arbitrary text. For files created via **O/B References–>Write (All) To File**, this will show the interface item's and onboard item's software types and engineering units resp. state code lists (as applicable) to allow identification of mismatches for the user.

Entries with a model interface item name that does not refer to an actual model interface item are ignored, undefined onboard references (i.e. references to onboard items that cannot be accessed resp. are not visible in the current CCU scope) may be adapted interactively, see section 7.4.7.8.

The pop–up menu command **Variable–>Reset Onboard Reference** provides the possibility to reset (i.e. delete or clear) the association between the selected top level interface item and onboard item.

If the Composite Editor is located on top level, two commands are available allowing to reset multiple onboard references at once. The command **O/B References–>Reset All** is available if no item is selected and may be used to reset all of the model's references to onboard items. The command **O/B References–>Reset** is available if a grouping link at the model (i.e. top level) interface is selected. It allows to reset the onboard references of the model interface items belonging to the selected logical grouping.

It is possible to generate a listing of the model interface items and the referenced onboard items in an ASCII file. The format is the same as described above, so the file may later be used to set onboard references from.

Two pop–up menu commands are provided which are available if the Composite Editor is located on top level. If no item is selected, the command **O/B References–>Write All To File** allows to list the complete model interface. The command **O/B References–>Write To File** is available if a grouping link at the model (i.e. top level) interface is selected. It allows to list those model interface items belonging to the selected logical grouping. In both cases the listing comprises also those model interface items that have no onboard reference. Each entry has a comment that gives information about the type of the model interface item and of the referenced onboard item, if there is one.

7.4.7.1.23 Selecting a Connection

Point with the cursor to the connection you want to select. Click the left button. Note that the end of each of these lines is indicated by a black dot. This informs you that the connection is selected.

Note that there can only be one connection selected at a time. Additional selection is not provided.

7.4.7.1.24 Deselecting a Connection

To deselect a connection simply move the cursor inside the graphic subview to an empty location and click the left button.

Another way is to point the cursor to the selected connection and click the middle button. But before doing so you have to make sure that the pointer button is activated.

7.4.7.1.25 Removing a Connection

Select the connection you want to remove. Execute the menu command **cut**. The connection disappears. The affected i/o–items are in unconnected state.

The basic menu command **undo** allows you to get back the last removed connection.

7.4.7.1.26 Disconnecting an i/o–item

You know that one i/o–item can be connected to a group of i/o–items. For example think of an atomic output connected to several inputs. If you cut a connection all affected i/o–items become disconnected. So cutting is not the right way if you want to disconnect only one of the inputs.

To do so, select the appropriate i/o–item and press the left mouse button. Execute disconnect. See that only the connection lines from the selected i/o–item up to the next junction disappear.

7.4.7.1.27 Moving Connection Lines

When you connect i/o–items the MDE creates connection lines by default.

You can move each of these lines. Horizontal lines can be moved up and down, vertical lines can be moved to the right resp. to the left. Select a connection to see the end point of each line indicated by a black dot.

Point the cursor to the particular line that you want to move. Press the left button. The line starts blinking. Move the cursor around. Note that, as the cursor moves, the line follows accordingly. Probably one or two additional lines are used to connect the moved line to the rest of the connection. This rubber band function tells you the currently suggested solution to substitute the original line.

When you are satisfied, release the left button. The last provided solution is inserted into the connection substituting the original line.

7.4.7.1.28 Splitting Connection Lines

It may be possible that you want to move only a part of a longer connection line. To do so, you have to split this line into two separate lines. Select the connection and press the left mouse button and choose the item **split**. Note that the message subview shows the following text: *Please click any button at the point where you want to split.*

So move the cursor over the particular line that you want to divide into two lines. Click the left button. The connection stays selected showing the new line's end points.

The basic menu command **undo** allows to get back to the last unsplit connection line.

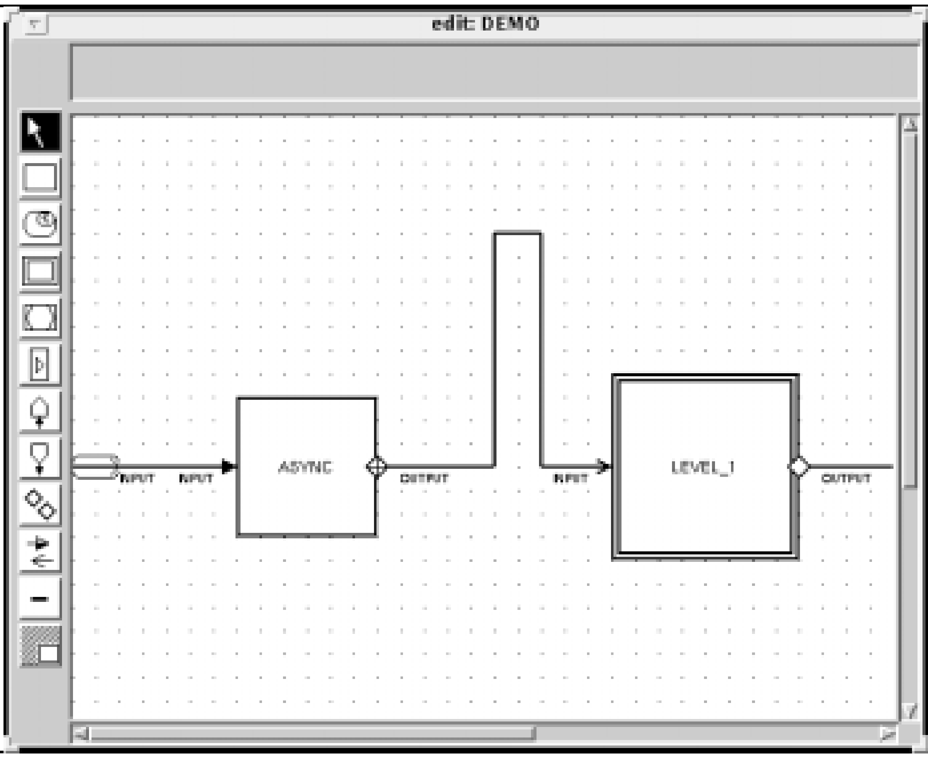


Figure 7–53 : *Moving a splitted connection line*

7.4.7.1.29 Manipulating connected objects

Almost every operation appropriate to block objects and i/o–items can be performed when these objects are connected, too. This comprises moving, resizing, cutting, copying and pasting. The affected connections will become updated automatically.

7.4.7.1.30 Logical Groupings

A logical grouping is a bidirectional bus that combines an arbitrary number of signals. Grouping entries provide the interfaces to a logical grouping. The interfaces to the signals contained in the logical grouping are the inputs and outputs of the grouping entries.

A completely defined signal has one input to a logical grouping and a number of outputs from the logical grouping. Each of these items representing the signal has the same name (the signal's name). Note that each signal can be interfaced only once at a particular grouping entry.

Create two grouping entries and select the associated grouping links. Execute **connect**. Create a composite function block and assign a grouping link to it. Select the grouping link and one of the grouping entries' grouping link and execute **connect** again. Step inside the composite function block and note that the grouping link is visible inside, too. Now create another grouping entry and connect its grouping link to that of the composite function block seen from inside.

Until now, the logical grouping does not contain any signals. Assign inputs and outputs to the grouping entries. Note that each of these items initially gets a name that is unique in the entire logical grouping. Each of these items represents a signal that is identified by its name. The initially created signals are incomplete.

Select an output at a grouping entry and press the right mouse button. Execute the menu command **connect signal**. A pop–up menu appears providing a list of signal names you can connect the output to. If you choose one of them, the output is renamed. The same works if you select an input of a logical grouping. Note that the provided list of signals is filtered with respect to type compatibility.

To rename a signal, select the i/o–item and execute **rename signal**. Type in the new name. Note that the name is accepted only if it does not already identifies another signal in the logical grouping. The entire signal is renamed (i.e. all i/o–items of grouping entries representing the signal are renamed).

The command **break signal** allows to disconnect an input or output of a grouping entry from a signal. In fact, you have to create a new incomplete signal that is represented exclusively by the selected i/o–item. A text field appears inviting you to specify the new signal's name. The name is accepted only if it does not already identifies another signal in the logical grouping.

To connect an atomic output to atomic inputs via a logical grouping, you have to connect the atomic output to a signal (i.e. an input to a logical grouping). Then you have to connect the atomic inputs to the same signal (i.e. outputs from the logical grouping that have the same name than the signal's input).

You can get a list of all signals contained in a logical grouping by selecting a grouping entry or a grouping link associated to a composite function block and executing the menu command **show signals**.

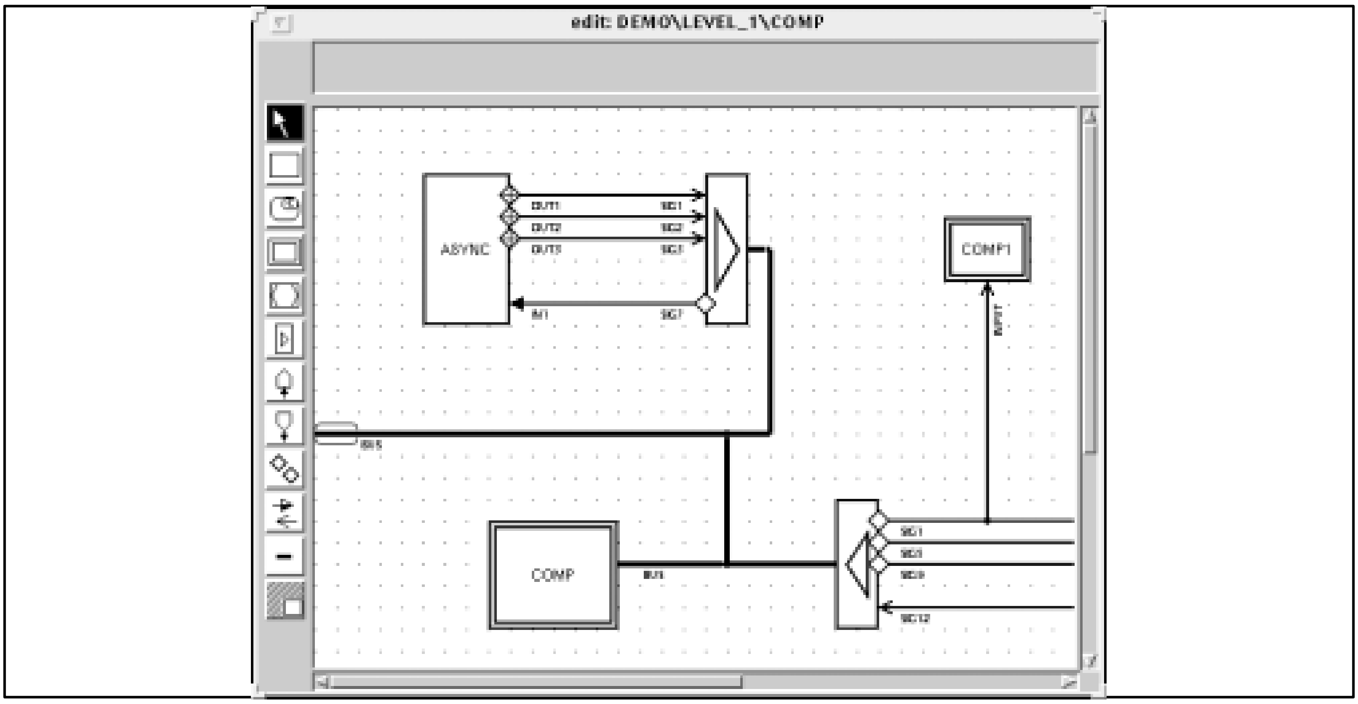


Figure 7–54 : *A Composite Editor showing an example of a logical grouping.*
Note that ASYNC\OUT1 is connected to COMP\INPUT via logical grouping (signal SIG1)

If you connect two logical groupings, the system checks the signals of both groupings. It’s not allowed to have two signal inputs with the same name in a logical grouping. The signals are also checked on type compatibility. If errors are found, an error window is opened providing error descriptions; the operation of connecting the two logical groupings becomes aborted.

7.4.7.1.31 Global Symbols

Global symbols provide another convenient way of connecting data sources (top level inputs, atomic outputs) to data sinks (top level outputs, atomic inputs). A global symbol consists of one definition point and a number of reference points that are identified by the same name (the global symbol’s name).

The definition point of a global symbol has to be defined either ***inside the same*** *or on a* ***higher level*** *composite function block than the reference points of this global symbol.*
This is called the visibility rule for global symbols.

Create a number of definition points and reference points. Select one of the reference points and press the right mouse button. Execute the menu command **connect**. You get a list of the names of all definition points that you can connect the reference point with. Choose one of them and see that the reference point is renamed to the selected name.

Create a composite function block and step into it. Create further reference points and connect them to the previously defined definition points.

To disconnect a reference point from a definition point, select the reference point and execute the menu command **break reference**. Automatically a new name is generated for the selected reference point that does not reference a visible definition point (i.e. a definition point defined in the same or a higher level composite function block).

You can rename an entire global symbol by selecting its definition point and executing the menu command **rename**. A text field appears allowing you to type in the new name. If the name is ok (does not already identify another object), the definition point and all connected reference points are renamed.

If a definition point is selected, the menu you obtain by pressing the right mouse button provides a command **show references**. On executing this command, you get a list of the pathnames of all connected reference points (including their type).

To connect an atomic output to atomic inputs via global symbol, you have to connect the atomic output to the definition point. Then you have to connect the atomic inputs to the global symbols' reference points.

The list of visible definition points you get when you want to connect a reference point to a definition point provides only those definition points that are type compatible. However, it may be possible that the types of data source and data sinks connected to a global symbol differ. For example, if you cut a definition point, its associated reference points may become associated to a definition point with the same name on a higher composite hierarchy level (if there is one); this may lead to type incompatibilities in the global symbol.

If a definition point or a reference point is selected, the pop–up menu obtained by pressing the left mouse button provides a command **unify type**. On executing **unify type**, a list of all types occurring in the global symbol appears. If you choose one of them, the type of all connected data items is set to the selected one.

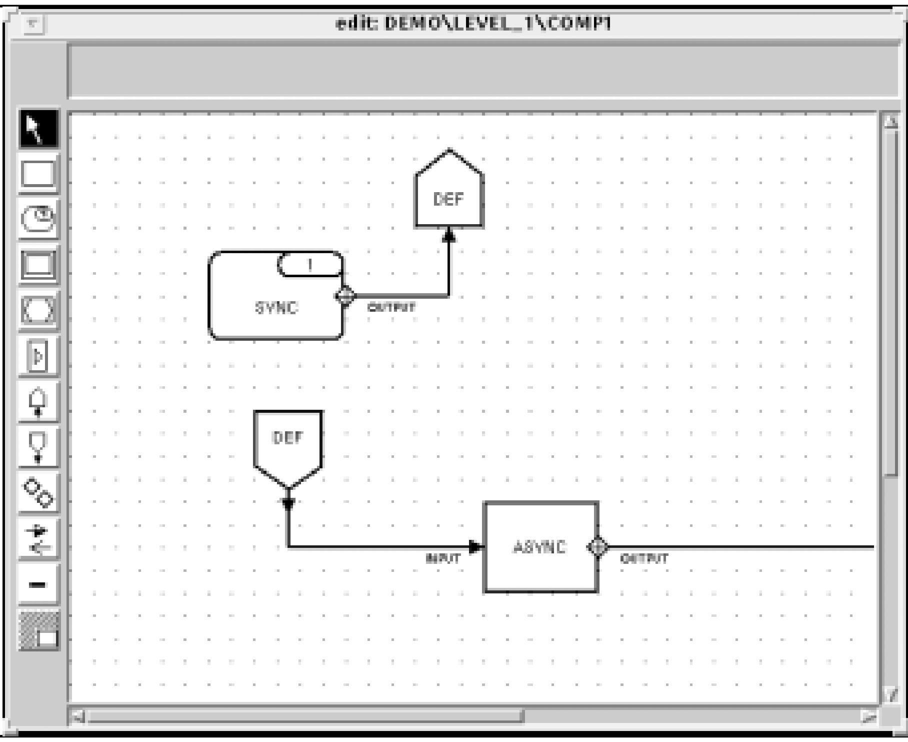


Figure 7–55 : *A Composite Editor showing an example of a global symbol. Note that SYNC\OUTPUT is connected to ASYNC\INPUT via global symbol DEF.*

7.4.7.1.32 Editing an Atomic Function Block

Atomic function blocks can be implemented by AIL–code (AIL–code is defined as a subset of the Ada language) or by decision table.

Select an asynchronous or a synchronous function block. Execute the menu command **edit**. A submenu appears allowing you to open either an AIL–Editor or a Decision Table Editor (submenu items **AIL code** resp. **decision table**).

If you try to open an AIL–Editor (resp. Decision Table Editor) on an atomic function block that is implemented by decision table (resp. AIL code), a confirmer appears warning you that the current implementation may become lost.

The use of following Ada features is **not allowed** within AIL code:
access types, package declaration, Ada tasking, address clauses

For more information about data types refer to section 7.4.7.1.37.2 and chapter 7.4.13.

For more information about mathematical constants and routines refer to chapter 7.4.12.

7.4.7.1.33 Changing the Grid

All operations including the determination of objects' positions (moving, pasting) are restricted to the current grid. The default grid is indicated by the dot pattern of the graphic area.

When there are lots of connections and you want to move some connection lines you may find out that the default resolution is not fine enough. So it may be impossible to find a suitable place that is not occupied by other connection lines. Choose the basic menu command **grid align**. A submenu appears listing the available grid values: **default**, **1/2** (* default), **1/4** (* default). Choose a finer grid and try some moving operations.

Note that the dot pattern indicates the default grid, not the current grid.

Note that i/o–items minimal spacing is restricted to default grid to avoid overlapping.

7.4.7.1.34 Changing the Size of a Block's Inside View

The default size of the inside view of a composite function block is A5. While editing you may find out that you want to enlarge the size.

Choose the basic menu item **size**. A submenu appears listing the available sizes: **A5, A4**, **A3** and **A2**. Choose the desired size and continue editing.

Note that a reduction of size affords that a part of the document has to be cut off. This part has to be empty. Otherwise the operation will be aborted indicated by a message displayed in the message subview

7.4.7.1.35 Searching for an Object

Choose the basic menu item **find**. A submenu appears asking you if you want to search for a function block, constant block, global symbol or for an i/o–item. Execute the submenu command **function** or **i/o**.

If you execute **function**, a pop–up menu appears listing all function blocks, constant blocks, global symbols defined on the actual composite level. The objects are identified by their names. Choose the object you want to find. See that the object becomes selected. If it is not already visible inside the graphic subview, the subview will be scrolled automatically.

The submenu command **i/o** works in a similar way. The appearing pop–up menu lists the names of all i/o–items defined at the currently edited composite function block (i.e. i/o–items to the next higher level seen from inside).

7.4.7.1.36 Changing a Function Block's Type

Select an asynchronous function block and press the right mouse button. Choose **change type**. A submenu appears listing the possible function block types, in this case **synchronous** and **composite**.

Similarly a synchronous function block can be converted to an asynchronous or a composite function block; a composite function block can be converted to an asynchronous or a synchronous function block.

Note that you may loose some of the information belonging to the selected object. If you convert an atomic function block to a composite function block, the implementation gets lost as well as the activation characteristics of the associated inputs, for example. On the other hand, if you convert a composite function block to an atomic function block, the contents of the composite can't be kept.

7.4.7.1.37 Defining Variables

If an output or an input is selected, the pop–up menu provides an item **variable**. If the cursor is positioned over **variable**, a submenu appears listing the details that can be specified to define the variable.

Each variable is to be defined by a type . On executing the submenu command **type**, a pop–up menu appears providing a list of possible types. The contents of the list of possible types depends on wether you selected a Top Level I/O (Inputs/Outputs connected to the border of the Top Level function block) or I/Os connected to function blocks within the model.

Variable Type	***Top Level I/O***	***Function Block I/O***
Unsigned_Byte	*X*	*X*
Signed_Byte	*X*	*X*
Unsigned_Short_Word	*X*	*X*
Signed_Short_Word	*X*	*X*
Unsigned_Integer	*X*	*X*
Signed_Integer	*X*	*X*
Real	*X*	*X*
Long_Real	*X*	*X*
Boolean	*X*	*X*
State_Code	*X*	*X*
Pulse	*X*	*X*
Burst_Pulse	*X*	*X*
Complex	*not available*	*X*
Time	*not available*	*X*
Long_Duration	*not available*	*X*
Vector	*not available*	*X*
Matrix	*not available*	*X*
Record	*not available*	*X*

Figure 7–56 *: A list of all available data types*

Figure 7–56 provides a list of the available types. The latter six types are available only within the model. Onboard types (top level inputs, top level outputs) have to be one of twelve types listed first.

Depending on the selected i/o–item (atomic input/output, composite input/output, model input/output) and the i/o–item's type the submenu that appears on executing **variable** provides the appropriate commands to define the variable.

If the variable type is VECTOR, MATRIX or RECORD separate input windows appear which allow to define the selected data type.

If the type is defined, you can specify the initial value (submenu command **initial value**). Depending on the variable's type, a text field or a pop–up menu appears allowing you to designate the initial value. The value is checked on correctness, i.e. the initial value setting includes a type check.

The write access can be specified by executing the submenu commands **enable write access** or **disable write access** (set resp. reset the write access marker).

Note that all outputs are automatically created with the write access marker set.

If the variable type is REAL or LONG_REAL, the menu provides a field **engineering unit**. A text field allows you to specify the engineering unit.

Connections can only be established between type compatible i/o–items. In general, type compatibility is given if the type identifier is the same.

7.4.7.1.37.1 STATECODE,VECTOR,MATRIX,RECORD as I/O

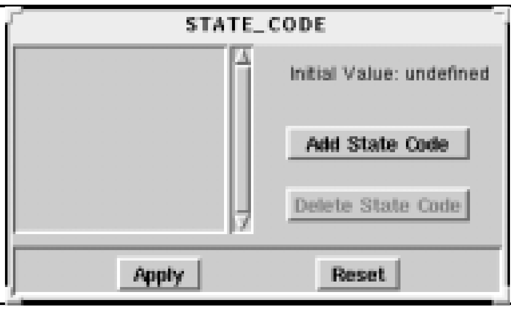


Figure 7–57 : *The empty state code definition window*

Creating a state code variable

- Select the desired input/output by clicking on it with the left mouse button.
- Press and hold the right mouse button. The related pop–up menu appears.
- Move the mouse pointer to the **variable–>type** command and release the mouse button. A list of all available data types appears.
- Move the mouse pointer to the entry **STATE_CODE** and press the left mouse button.
- The definition window pops up (see Figure 7–57). Press the **Add State Code** button.
- Type the new state code in the input line. (see Figure 7–58) and press the **Accept** button. The state code is listed in the list subview in Figure 7–57.

Note that the state code is locked up in double quotes , the string has to be filled with blanks to fit into the 8 characters limit.

- Repeat the preceding steps to enter more state code definitions.
- Press the **Apply** button in the state code definition window.

Figure 7–58 : *The state code input window*

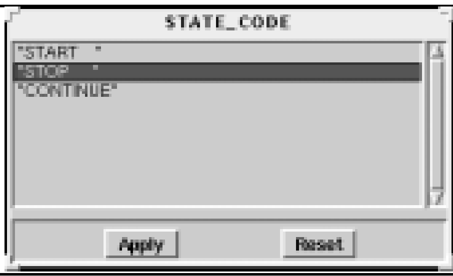


Figure 7–59 : *The initial value selection window for state code variables*

Setting the initial value for state code parameters

- Select the desired input/output by clicking on it with the left mouse button.
- Press and hold the right mouse button. The related pop–up menu appears.
- Move the mouse pointer to the **variable–>initial value...** command and release the mouse button. The initial value selection window (see Figure 7–59) appears.
- Select the desired initial value by clicking on the name in the list. The selected value becomes high-lighted.
- Press the **Apply** button.

For the data types vector, matrix and record some additional rules appear.

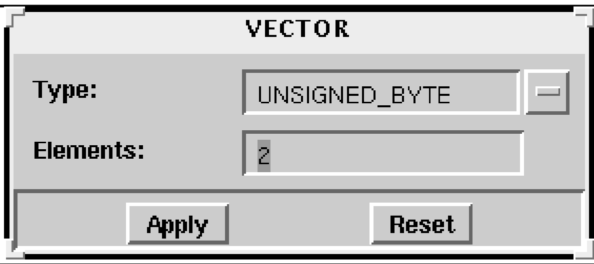


Figure 7–60 : *Definition window for a variable of type vector*

After definition of a variable of type vector the window as shown in Figure 7–60 appears. The definition window for vector types shows the default number of elements for a vector (which is at least 2). Type the number of vector elements you need, then press the **Type** button. A predefined list of available data types for the vector elements pops–up as shown in Figure 7–61. Select one of the data types, then press the **Apply** button. In the example above a vector variable with 2 elements, both of type REAL is created.

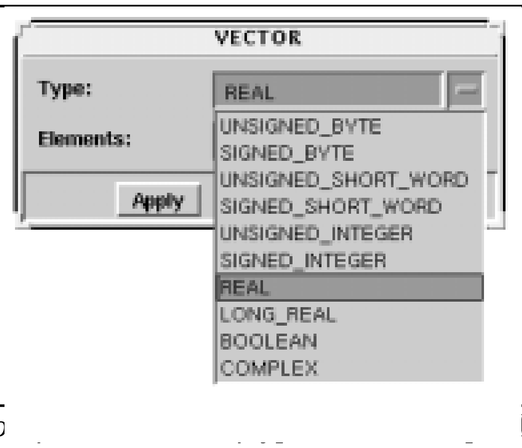


Figure 7–61 : *Definition windo ... lefined list of data types.*

Note that you can choose only one data type for all elements of a vector.

After definition of a variable of type matrix the matrix definition window appears. Change the number of rows and columns accordingly. The press the **Type** button. Select the data type for the matrix elements from the predefined list then press the **Apply** button.

Figure 7–62 : *Definition window for a variable of type matrix with predefined rows and columns.*

Note that you can choose only one data type for all elements of a matrix.

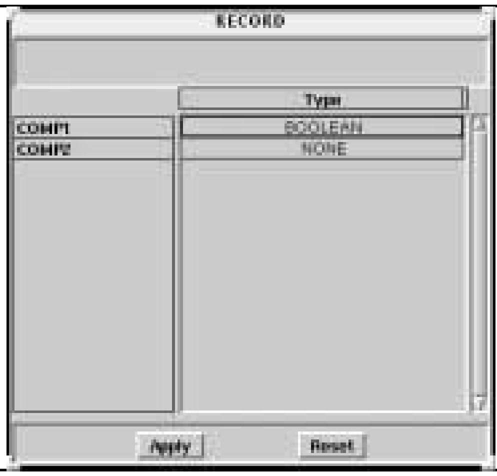


Figure 7–63 : *Definition window for a variable of type record.*

Note that the data type record is only a graphical combination of different variables.

Figure Figure 7–63 shows the definition window for the record data type. Select the field COMP1. This is the default name for the first component. The pop–up menu command **rename** allows to set a new name for this component. With the command **insert component –> preceding** or **insert component –> succeeding** you can add further components before or behind the selected one. The pop–up command **delete** removes the selected component.

With no component selected the pop–up menu provides the commands **undo** and **add component**.

To change the type of a component select the field in the column Type. Select the data type from the list of predefined types.

Note that each component in the record can be of different data type.

For more information about data types refer to section 7.4.7.1.37.2.

7.4.7.1.37.2 The AIL data types

This section describes the AIL data types used for the implementation of AIL code and the connection between the function blocks.

Following informations will be listed:
Data Type, the internal representation, the range, further informations i.e. examples if necessary

For the detailed Ada specification of all available AIL data types and their available functions refer to chapter 7.4.13.

UNSIGNED_BYTE

internal representation: 8 bit unsigned integer
range: 0 to +255

SIGNED_BYTE

internal representation: 8 bit unsigned integer
range: –128 to +127

UNSIGNED_SHORT_WORD

internal representation: 16 bit unsigned integer
range: 0 to +65535

SIGNED_SHORT_WORD

internal representation: 16 bit unsigned integer
range: –32768 to +32767

UNSIGNED_INTEGER

internal representation: 32 bit unsigned integer
range: 0 to +4294967295

Note that in the current implementation the upper limit is +214748367

SIGNED_INTEGER

internal representation: 32 bit unsigned integer
range: –2147483648 to +2147483647

REAL

internal representation: 32 bit IEEE single float
range:

LONG_REAL

internal representation: 64 bit IEEE single float
range:

BOOLEAN

internal representation: enumeration type
default values: TRUE, FALSE

STATE_CODE

internal representation: 8 character string

State code naming conventions are:

– the first character must be a letter
– any subsequent characters must be letters, digits, or the underscore ('_')
– two underscores cannot occur together, nor can a state code end with an underscore
– all state codes will be in capital letters only

PULSE

internal representation: **BOOLEAN**

REMARK:
A pulse can be set to TRUE during model run–time via MOCS, it will be automatically reset after the execution of the time frame the pulse is associated with.
Only one pulse will be generated.

BURST_PULSE

internal representation: **UNSIGNED_INTEGER**

REMARK:
The burst pulse number can be set during model run–time via MOCS, it will be automatically reset after the execution of the time frame the burst pulse is associated with.

COMPLEX

internal representation: re: **LONG_REAL**, im: **LONG_REAL**

Note that you can choose between GAUSS or POLAR notation in the input window display.

TIME

internal representation:
- year: INTEGER, range: 1901 .. 2099,
- month: INTEGER, range: 1 .. 12,
- day: INTEGER, range: 1 .. 31
- seconds past midnight: DURATION

Note that the user enters the seconds past midnight in a simplified input format:
- *hour: INTEGER, range 0 .. 23,*
- *minute: INTEGER, range 0 .. 59,*
- *second: float, range 0.00 .. 59.99*

EXAMPLE: 1.1.1993 12:17:59.12

LONG_DURATION

internal representation: DURATION
range: –86400.00 sec to 86400.00 sec

Note that in the current implementation the internal representation is fixed point

VECTOR

internal representation: 1–dimensional array of a scalar type
range: max 255 elements

For the vector elements following scalar data types are allowed:

UNSIGNED_BYTE,
SIGNED_BYTE,
UNSIGNED_SHORT_WORD,
SIGNED_SHORT_WORD,
UNSIGNED_INTEGER,
SIGNED_INTEGER,
REAL,
LONG_REAL,
BOOLEAN,
COMPLEX

MATRIX

internal representation: 2–dimensional array of a scalar type
range: max 255 elements **in total**

For the matrix elements following data types are allowed:

see above VECTOR

RECORD

is a **graphical** association of inputs/outputs.

internal representation: is the internal representation of the record elements
range: The number of elements in a record should be less than 200. The accurate number can not be given because it depends on the total number of inputs/outputs implemented.

For the record elements following data types are allowed:
see above VECTOR

- *In AIL code you have access to the element of a record via record name – underscore – element name.*
  EXAMPLE:
  The name of the record is CABLE with the elements POWER, STATUS and VOLTAGE.
  You can access the three values via CABLE_POWER, CABLE_STATUS and CABLE_VOLTAGE.

7.4.7.1.38 Performing a Rule Check

The basic menu command **check** allows to check a simulation model on correctness. A window appears inviting to specify the checks to be performed by selecting the corresponding buttons.

Figure 7–64 : *The Rule Check Parameters window*

To check the model on the existence of asynchronous loops, select the button **Check: Asynchronous Loops**. The completeness and correctness of the model's global symbols can be verified by selecting **Check: Global Symbols**. To check whether all atomic and composite function blocks are implemented, select **Check: Implementations**.

The model's variables can be checked on type definitions (select **Check: Type Definitions**) and on initializations (select **Check: Initializations**). Unconnected interface items are found by selecting **Check: Unconnected I/F Items**.

On pressing the **Apply** button the checks are performed, possibly resulting in the generation of warning and/or error messages. It is possible to suppress the display of warnings by selecting the button **Warnings: Suppress**. Otherwise, if warnings should be displayed, select **Warnings: Show**. If errors (and/or warnings) occur, an error window is opened providing detailed information.

7.4.7.1.39 Saving the Model

To save the current state of your work, execute the basic menu command **save**. The cursor changes shape to that of a pen indicating that all the modifications are written.

Impact on executable simulator kernel and state vectors :

For proper execution of a simulation model, sufficient consistency between the generated runtime kernel and the model source is required. Thus, when saving the edited model, the runtime kernel and all associated state vectors are automatically deleted by the model editor in case it has identified significant changes in the model semantics. Significant changes in this context are e.g.:

- creation, deletion, renaming, change of frame or type of function blocks
- creation, deletion, renaming, change of activation or data type of FB interface items

Minor changes, which will not result in deletion of kernel and state vectors, in this context are e.g.:

- Graphical layout changes (moving of FBs, FB I/Os or connections)
- *Any changes in atomic FB source code.* This is in order to avoid deletion of the runtime files due to accidental changes in some atomic FB source code, although it also implies that the atomic source code shown in the model observer during simulation execution may differ from the actual behaviour of the current runtime kernel.
- *Changed On–Board item references.* Since compatibility of top–level interface items and their connected on–board items can in general not be determined by CSS MDE, changed on–board references will keep the model executable. It is left to the responsibility of the model developer to execute the operation **Configure –> Adaptation System** to ensure proper configuration of the project specific adaptation system, including the applicable consistency checks.

7.4.7.1.40 Model Compilation and Simulator Kernel Configuration

7.4.7.1.40.1 Compiling atomic function blocks

The basic menu command **compile** allows to compile one or more atomic function blocks separately. You can select the atomic functions blocks to be compiled on the current level. If you select a composite function block all function blocks on lower levels in this composite are compiled. If there is no function block selected, all atomic function blocks in the model are compiled.

A window appears allowing to specify the compilation parameters. The **Architecture** button allows to select the target architecture. The **Host** button allows to select the target host on which the compilation shall be performed. The architectures and host names provided correspond to the specifications in the system's CSS configuration file.

Figure 7–65 *: The Compilation Parameters window*

To reduce the time consuming compilation phase, incremental compilation is provided. Incremental compilation means only those atomic function blocks which have been edited since the last compilation will be compiled. Select the button **Compile: Updated** to perform incremental compilation. By selecting **Compile: Forced** all selected atomic functions blocks will be compiled.

In rare conditions, the generated Ada source files corresponding to the atomic function blocks may be corrupt. If the button **Sync Atomic Source Files** is selected, the system writes, prior to compilation, the source code of all atomic function blocks once again to the file system. However, this operation, that may be time consuming for bigger models, is necessary only in exceptional conditions (e.g. if someone has manipulated the source files in the file system).

When both **Compile: Updated** and **Sync Atomic Source Files** is selected, the system will also first purge the complete architecture–specific file system branch for the model, including executable and Ada library, which may occasionally be necessary to recover from a crashed compilation.

Prior to compilation, an MDE–GL rule check, possibly resulting in the generation of warning and/or error messages, will be performed. It is possible to suppress the display of warnings by selecting the button **Warnings: Suppress**. Otherwise, if warnings should be displayed, select **Warnings: Show**.

On pressing the **Apply** button, a confirmation window may appear advising to save the model if there are any pending modifications. If saving is not confirmed, the operation is aborted. Then the rule check is performed, and, if any errors are found, the operation is aborted. An information window is opened showing the errors (and/or warnings) generated by the rule check, if there are any. Another information window provides the compilation status and lists the warnings and/or errors which occur during compilation.

The compilation information is additionally stored in the Unix file system in a directory which is determined by the environment variable CSS_LOG_DIR . The file is marked with the extension .COMPILATION_REPORT_TXT . The part of the name preceeding the extension matches the same patterns for log and archive files (refer to section 7.4.7.9.4 for more information on names of log and archive files).

7.4.7.1.40.2 Simulator Kernel Configuration

The basic menu command **configure–>simulator kernel** compiles the model's atomic function blocks, configures an executable simulator kernel (i.e. generates and compiles a runtime system and links the atomic function blocks with the runtime system) and generates the initial simulation state.

A window appears allowing to specify the compilation parameters (target architecture, target host, enable/disableincremental compilation, suppress/show warnings, sync atomic source files) as described in the previous section.

On pressing the **Apply** button, a confirmation window may appear advising to save the model if there are any pending modifications. If saving is not confirmed, the operation is aborted. Then a complete rule check is performed. An information window is opened showing errors (and/or warnings), if there are any. If there are any errors found, the operation is aborted.

While simulator kernel configuration is performed, the system provides progress information in an information window. You are informed whether the simulator kernel could be configured successfully or errors have led to breaking off the operation. Note that the action of simulator kernel configuration may be time consuming.

The configuration progress information is additionally stored in the Unix file system in a directory which is determined by the environment variable CSS_LOG_DIR . The file is marked with the extension

.COMPILATION_REPORT_TXT . The part of the name preceeding the extension matches the same patterns for log and archive files (refer to section 7.4.7.9.4 for more information on names of log and archive files).

If the simulator kernel configuration terminated successfully, you can start a simulation using MOCS.

Note that an initial simulation state is created during simulator kernel configuration. This simulation state sets the simulation to the start conditions.

The simulation state defines the start–up state of your simulation. If you are not interested in starting the simulation just from the initial state you can use MOCS to create more simulation states.

7.4.7.1.40.3 Adaptation System Configuration

If the model is intended to be used for H/W in the loop simulations, i.e. it's interface references onboard items in the database, the basic menu command **configure–>adaptation system** allows to generate new configuration files (one for each architecture a simulator kernel is configured for) for the Command and Measurement Adaptation System (CMAS). All onboard items referenced by the model are checked on completeness and consistency. If warnings or errors occur, an information window providing detailed information is opened.

Since definitions of onboard data and adaptation requirements (i.e. external interfaces of the simulator) are usually project specific, configuration for a project specific CMAS is supported by checking for the existence of an executable file **$CMAS_HOME/bin/common/start_db_server.** If present, this executable will be started instead of the internal standard configuration routine and given the responsiblity to generate a proper CMAS configuration file.
In this case, the rules for correct definition of referenced onboard end items as specified in section 7.4.11 will obviously be overruled by project specific definitions.

Each time the definitions of referenced onboard items have been modified in the database or the simulation model has been edited and a new simulator kernel has been configured, new configuration files have to be generated.

7.4.7.1.41 Printing Out a Document

Make sure that your laser printer is activated. Execute the basic menu command **print**. The documentation of the currently edited composite function block is printed out (graphic and i/o–list). The previous selection of more than one function block leads to a related documentation printout.
Note that the process of printing takes a little time.

For detailed information about documentation functions see chapter 7.4.7.7 (Documentation Generation Function).

7.4.7.2 Model Inspectors

For each function performed with a dedicated editor tool there is also an inspector tool provided. The inspector tools provide all functions specific to the editor except the function to make changes. For example see Figure 7–66 which shows the composite inspector window. Note that the tool button column on the left side is missing.

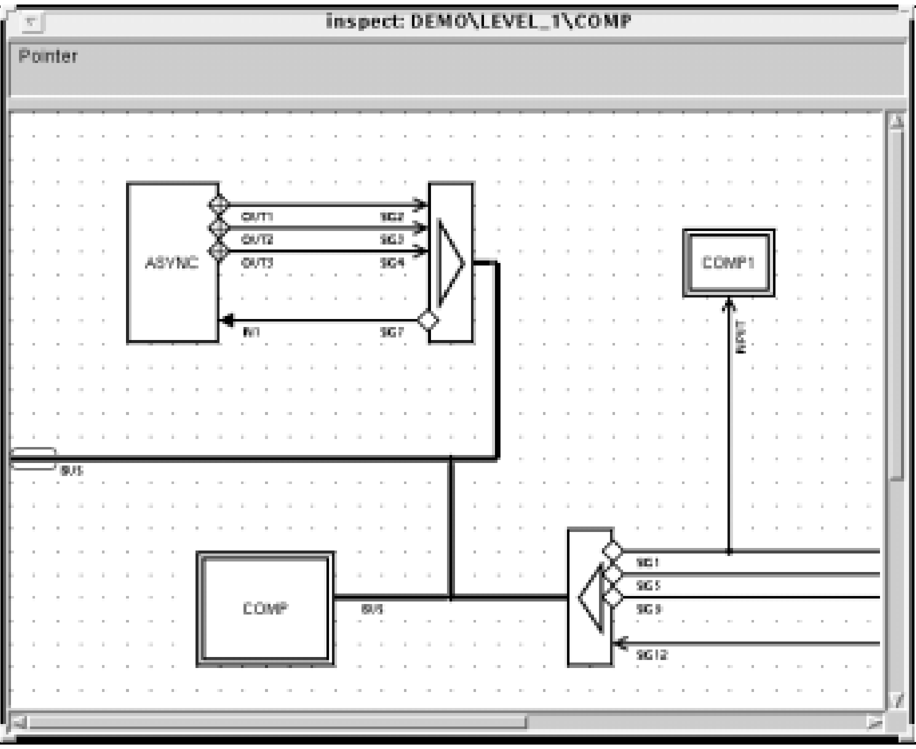


Figure 7–66 : *Composite Inspector User Interface*

The inspector tools are used to inspect items and to create documentation without changing the items.

7.4.7.3 Composite Interface Editor

The Composite Interface Editor runs in a separate MDE window. It may be activated from within the Data Base Browser or a Composite Editor. The tool allows to inspect and edit the interfaces of composite function blocks.

The Composite Interface Editor provides a graphic subview showing the composite function block including its interface and scrolling list subviews for all kinds of interface items (inputs, outputs, grouping links). A further scrolling list subview is used to present all signals of a selected logical grouping (if no grouping link is selected, this subview stays empty).

The interface items can be selected interactively either inside the graphic subview (selecting works in a similar way as in the Composite Editor) or in one of the scrolling list subviews. All these subviews provide selection sensitive pop–up menus to edit the interface.

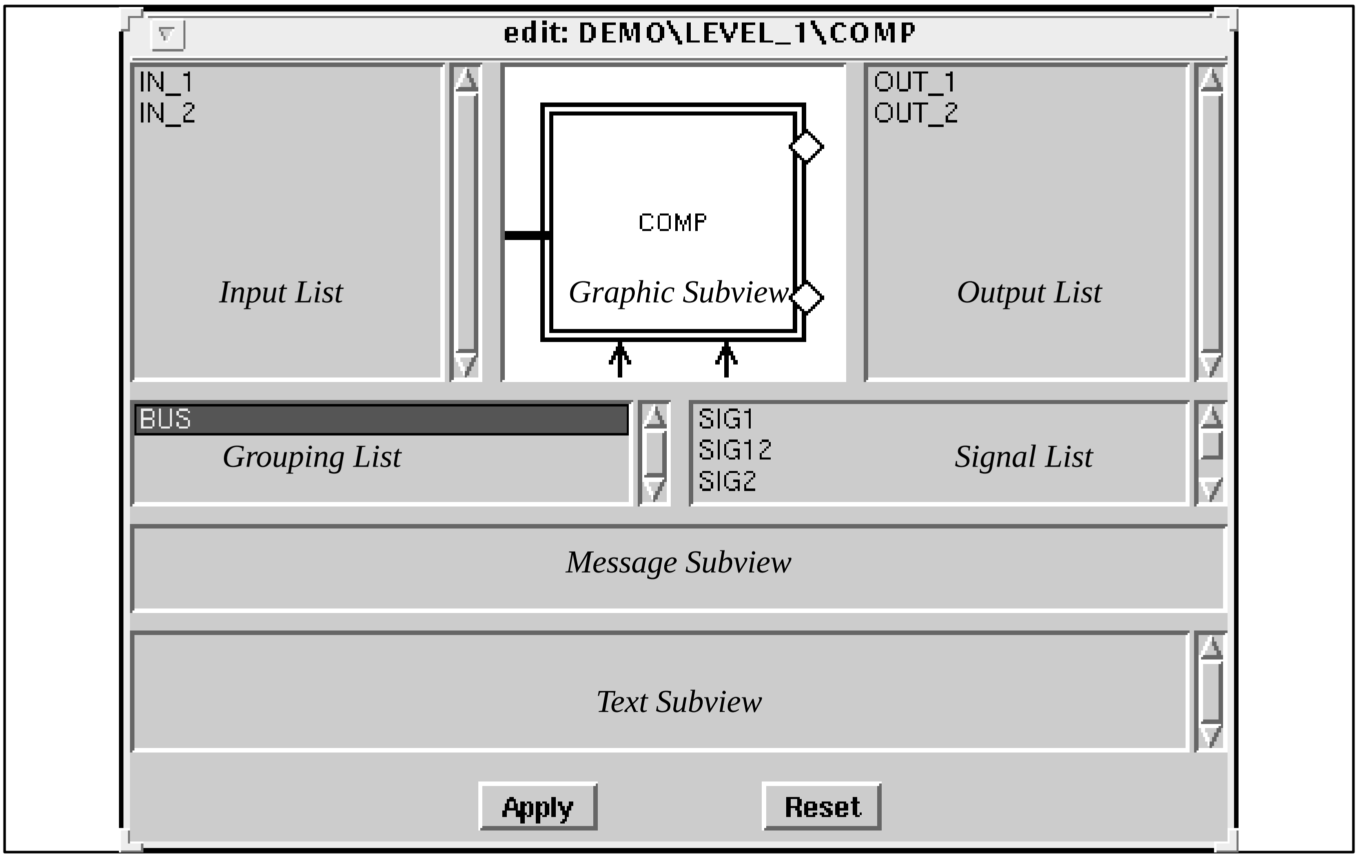


Figure 7–67 : *Composite Interface Editor user interface*

The main edit functions provided in the graphic subview and the scrolling list subviews are listed briefly:

* creating interface items (only inside the graphic subview)
* moving interface items (only inside the graphic subview)
* cutting interface items
* copying/pasting interface items (pasting only inside the graphic subview)
* copying the composite function block (to a Composite Editor)
* changing the attributes of interface items (e.g. type, initial value, etc.)

* generating and printing documentation

The Composite Interface Editor provides a text subview that can be run in inspect mode (read only mode). Usually if an interface item is selected, the text subview provides information about the selected item in read only mode.

If there is no interface item selected, the text subview shows the comment of the composite function block. In this case the view runs in edit mode allowing to change the comment. It provides the common text editing functions. These comprise:

* typing in text
* selecting text
* copying text (possibly cross text windows)
* cutting text

It is possible to copy the edited block to a Composite Editor. Furthermore the Composite Interface Editor allows to copy/paste interface items from and to a Composite Editor, another Composite Interface Editor, an Atomic AIL Editor or an Atomic Decision Table Editor.

The user may generate and print out documentation of the composite function block. The following outputs are provided:

* textual output describing a selected i/o–item (e.g. type, initial value)
* textual output of the composite function block's comment

User Interface

Figure 7–67 shows the user interface of the Composite Interface Editor. It provides scrolling list subviews (*input list*, *output list*, *grouping list and signal list*) and a graphical subview allowing interface item selection (*graphic subview*). Note that the signal list will be displayed after selection of an item in the grouping list. A read only text subview is used to provide textual feedback to the user (*message subview*). A text subview that can be run in read only mode as well as in write mode is used to give textual information of the selected interface item (*text subview*). When there is no interface item selected, the *text subview* runs in edit mode allowing to inspect/edit the comment of the composite function block.

Editing of the interface is done inside the *input list*, *output list*, *grouping list* and *graphic subview* via selection sensitive pop–up menus.

Saving the edited text within the text subview (comment) is done clicking on the **Apply** button, whereas via the **Reset** button one goes back to the last saved version.

7.4.7.4 Atomic Editors

The Atomic Editors are the tools to edit atomic function blocks. Since atomic function blocks can be implemented in two different ways (either by AIL code or by decision table), there are two kinds of Atomic Editors: the AIL–Editor and the Decision Table Editor.

You can open an Atomic Editor to edit an atomic function block from inside the Composite Editor. This chapter explains the different features of this tool.

7.4.7.4.1 Common

Using the Composite Editor, create several atomic function blocks. Assign several i/o–items to them. Establish connections between the i/o–items. Then select one of the atomic function blocks and execute **edit** and the submenu command **AIL code** to open an AIL Editor. Similarly, open a decision table editor on another atomic function block (submenu command **decision table**).

7.4.7.4.2 Components

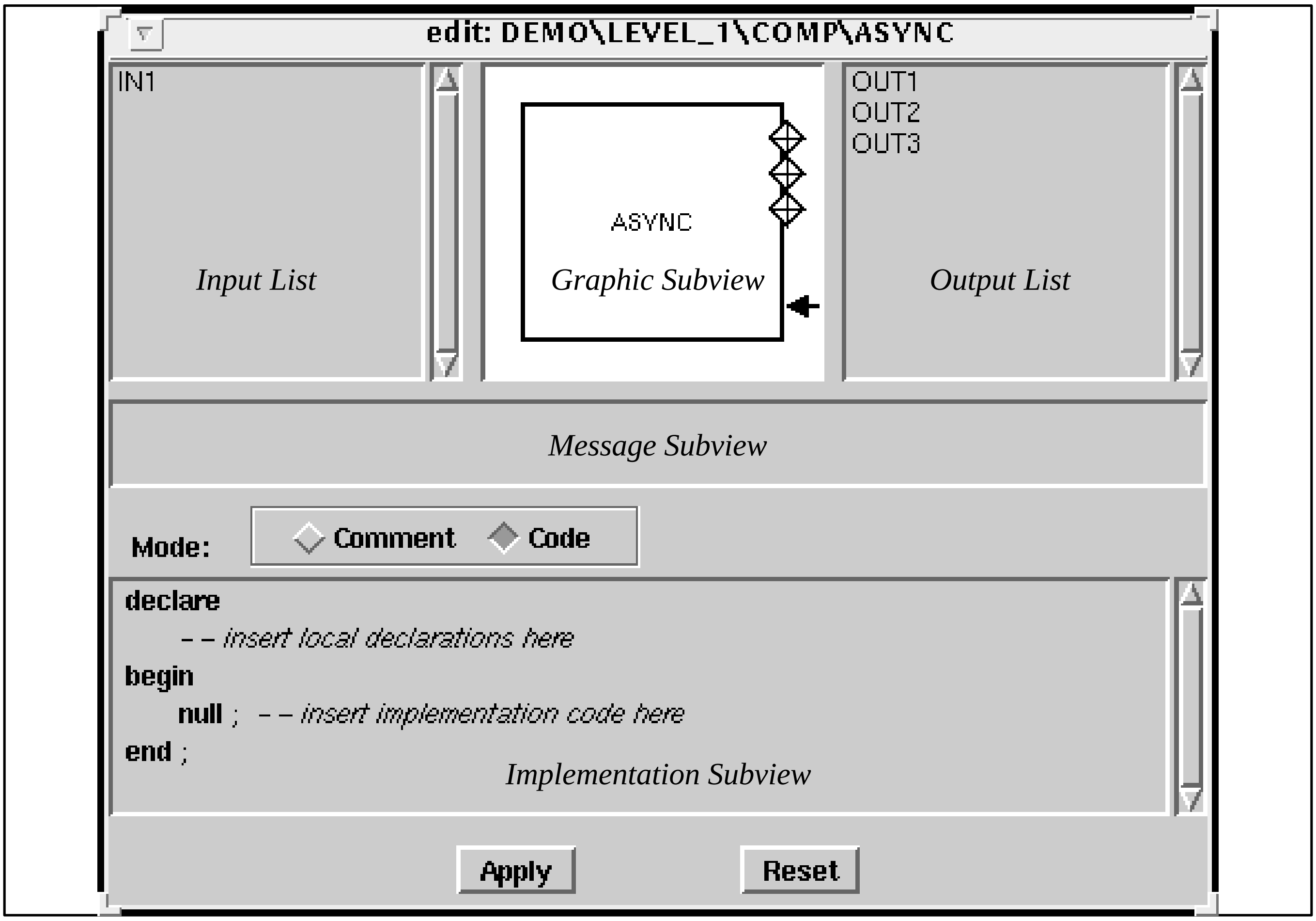


Figure 7–68 : *Components of the Atomic Editor*

7.4.7.4.2.1 The Label

The label of the atomic editor shows the pathname of the atomic function block.

7.4.7.4.2.2 The Graphic Subview

The graphic subview shows the graphical symbol of the atomic function block including all of its i/o – items. Note that connections are not displayed.

You can select the i/o – items quite similar like you used to do using the Composite Editor. So move the cursor over an output and click the left button. Note that accordingly the text subwindow changes its contents. It shows information about the selected i/o–item.

Note also that simultaneously in the output list the name of the selected formal out – parameter is highlighted. It is not possible to select more than one i/o–item at a time. To deselect move the cursor to a location so that it doesn't point to an i/o–item and click the left button.

You can edit the selected i/o–item by pop–up menu commands the same way as with the Composite Editor.

7.4.7.4.2.3 The Input and Output Lists

The input list shows the names of all inputs; the output list shows the names of all outputs. Because the subwindows may not be large enough do show the whole lists, a scrollbar is associated.

You can select an input or output by moving the cursor over its name and clicking the left button. Note that simultaneously in the graphic subview the appropriate i/o–item is selected. Information about the selected object is shown in the text subview.

To deselect, move the cursor over the highlighted name and click the left button. The i/o–item becomes deselected.

You can edit the selected i/o–item by pop–up menu commands the same way as with the Composite Editor.

7.4.7.4.2.4 The Message Subview

While editing the message subview is used to provide feedback. For example if you try to perform an illegal action you will be informed by a message indicating the mistake.

7.4.7.4.2.5 The Implementation Subview

Since atomic function blocks may be implemented either by AIL code or by decision table, the implementation subview of the AIL–editor is different from the implementation subview of the decision table editor.

However, in both editors, if there is an i/o – item selected, the implementation subview is used to provide information about this particular i/o – item. In this case you can't edit the displayed text.

7.4.7.4.3 Atomic AIL Editor

The AIL Editor provides a graphic subview showing the atomic function block including its interface and scrolling list subviews for the interface items (inputs and outputs).

The interface items can be selected interactively either inside the graphic subview (selecting works in a similar way as in the Composite Editor) or in one of the scrolling list subviews. All these subviews provide selection sensitive pop–up menus to edit the interface.

The main edit functions provided in the graphic subview and the scrolling list subviews are listed briefly:

* creating interface items (only inside the graphic subview)
* moving interface items (only inside the graphic subview)

* cutting interface items
* copying/pasting interface items (pasting only inside the graphic subview)
* copying the atomic function block (to a Composite Editor)
* changing the attributes of interface items (e.g. type, initial value, etc.)
* generating and printing documentation

The AIL Editor provides a text subview that can be run in inspect mode (read only mode). Usually if an interface item is selected, the text subview provides information about the selected item in read only mode.

If there is no interface item selected, the text subview shows the comment or the AIL implementation of the atomic function block. In this case the subview runs in edit mode. The user will be able to specify whether he wants to change the comment or the AIL implementation. This may be realized via related buttons (**Code** , **Comment**). The text subview provides the common text editing functions. These comprise:

* typing in text
* selecting text
* copying text (possibly cross text windows)
* cutting text

Since the text subview allows text copying cross windows it is possible to copy text from other text editors, i.e. at least this way the model developer is able to make use of other text editors (possibly file based) to implement an atomic function block.

It is possible to copy the edited block to a Composite Editor. Furthermore the AIL Editor allows to copy/paste interface items from and to a Composite Editor, another Atomic AIL Editor, an Atomic Decision Table Editor or a Composite Interface Editor.

The user may generate and print out documentation of the atomic function block. The following outputs are provided:

* textual output describing a selected i/o–item (e.g. type, initial value)
* textual output of the atomic function block's comment
* textual output of the atomic function block's AIL implementation

User Interface

Figure 7–69 shows the user interface of the AIL Editor. It provides scrolling list subviews (*input list*, *output list*) and a graphic subview allowing interface item selection (*graphic subview*). A read only text subview is used to provide textual feedback to the user (*message subview*). A text subview that can be run in read only mode as well as in write mode is used to give textual information of the selected interface item (*text subview*). When there is no interface item selected, the *text subview* runs in edit mode allowing to inspect/edit the comment or the AIL implementation of the atomic function block.

Editing of the interface is done inside the *input list*, *output list* and *graphic subview* via selection sensitive pop–up menus.

If there is no selected i/o – item, the text subview provides a text editor allowing to implement the atomic function block by AIL code.

Saving the edited text within the text subview is done by clicking on the **Apply** button, whereas via the **Reset** button one goes back to the last saved version.

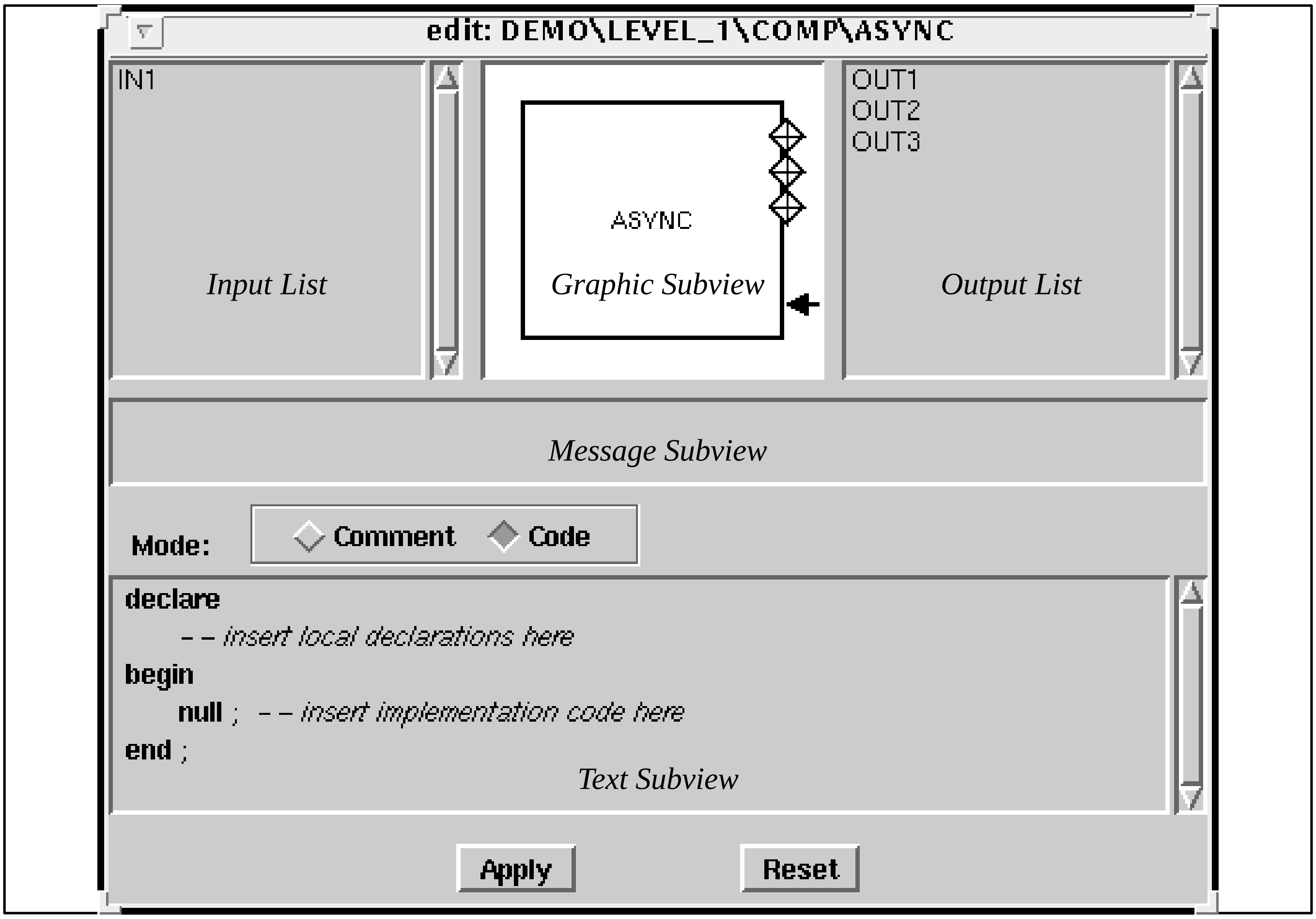


Figure 7–69 : *AIL Editor user interface*

7.4.7.4.3.1 AIL special features (hibernate, PULSE type parameter)

AIL is defined as a subset of the Ada language. (Refer to the Ada language standard for further information.) Following Ada features are out of the scope of the AIL editor:

- Ada tasking
- Address Clauses

Note that the length of a line is restricted to 80 characters.

There are some special features concerning PULSE type variables and the HIBERNATE function which will be described in detail here.

Set a synchronous FB to hibernate in AIL code

- Implement several lines of code which shall be performed before the hibernate command.
- Implement the condition under which the synchronous FB shall hibernate.
- Type **hibernate := true;** in the AIL code.
- Finish the AIL code correctly.

This implementation works as follows:

- in the beginning the synchronous FB is activated cyclically
- as soon as the conditions in which the hibernate command is embedded become true the hibernate command will be performed
- after finishing the AIL code the FB is in hibernate state and needs an activating event from another FB to start its cycle again

PULSE and BURST_PULSE parameter need to be treated in a different way than all the other parameters.

How to check wether a PULSE parameter is set

- Define an input of type PULSE.
- Implement the check as follows: **if <input> = PULSE_TRIGGERED then .... end if;**
  (<input> is any name)

How to pass a PULSE parameter forward to another FB

- Define an output of type PULSE.
- Implement the conditions on which the PULSE parameter shall be passed.
- Type in the AIL code **<output> := PULSE_TRIGGERED;** (<output> is any name)
- Finish the AIL code correctly.

Note that BURST_PULSE parameter are of type integer.

How to check wether a BURST_PULSE parameter is set

- Define an input of type BURST_PULSE.
- Implement the check as follows: **if (<input> /= BURST_PULSE_NOT_TRIGGERED) then .... end if;**
  (<input> is any name)

How to pass a BURST_PULSE parameter forward to another FB

- Define an output of type BURST_PULSE.
- Implement the conditions on which the BURST_PULSE parameter shall be passed.
- Type in the AIL code **<output> := <integer>;** (<output> is any name , <integer> is a number)
- *Note that <integer> determines the number of pulses in the BURST_PULSE parameter.*
- Finish the AIL code correctly.

7.4.7.4.3.2 Variables in the AIL code

Variables defined as inputs and outputs are used as variables in the AIL code too.

There are some special rules for the use of vector and matrix variables:

Each element or component must be written separately.

Example: Variable Price is a vector with 3 elements of type REAL.
Price(1) := 1.2;
Price(2) := Price(3);

Variable MAT is a matrix with 6 elements of type BOOLEAN
MAT(2,3) := TRUE;
MAT(1,1) := MAT(1,2);

Vector and matrix variables of the same type and size can be connected.

Example: Variable New_price is a vector with 3 elements of type real.
New_price := Price;

Variable MAT_X is a matrix with 6 elements of type BOOLEAN.
MAT_X := MAT;

Refer to the Ada reference manual for more information about operations of unconstrained array types. See also chapter 7.4.13.

There are some special rules for the use of record variables:

The name of a record component consists of the record name, an underscore and the component name.

Example: Variable Store has 2 components: much is of type REAL, enough is type BOOLEAN.
Store_much := 12398.12;
Store_enough := FALSE;

Record variables of the same type and size can not be connected.

Example: Variable N_Store has 2 components: much is of type REAL, enough is type BOOLEAN.
It is **not allowed** to write N_Store := Store.

To connect the variables each component has to be connected separately.

Example: N_Store_much := Store_much;
N_Store_enough := Store_enough;

Note that the record components can be treated like normal input/output variables of the same data type within the AIL code. This means all operations defined for a specific data type are available for the record component.

7.4.7.4.3.3 Restrictions

The following functions are not yet available in the current implementation.

- The necessary functions to use CSS type ”COMPLEX” are not yet implemented.

7.4.7.4.4 Atomic Decision Table Editor

The Decision Table Editor provides a graphic subview showing the atomic function block including its interface and scrolling list subviews for the interface items (inputs and outputs).

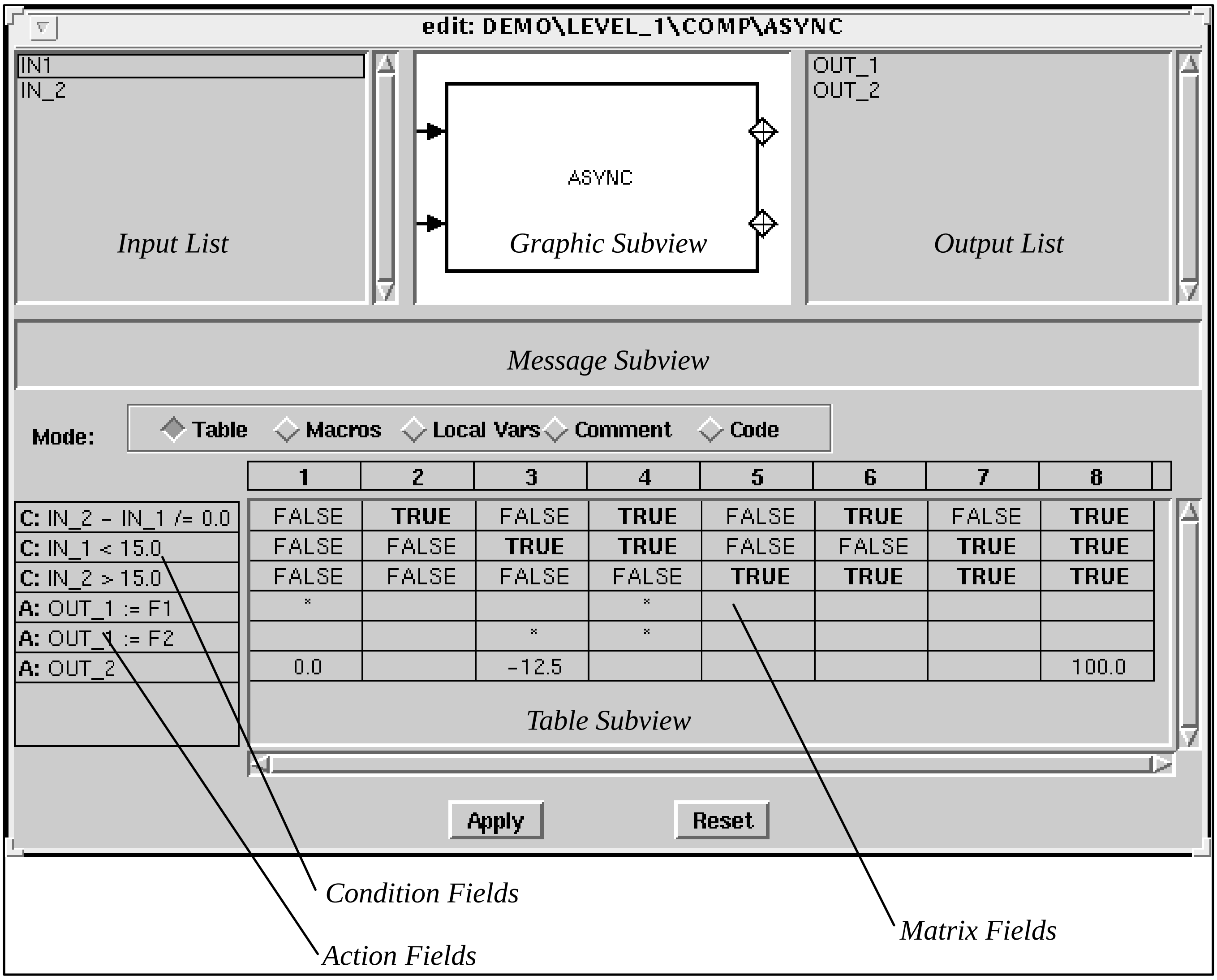


Figure 7–70 : *Decision Table Editor user interface*

The user may choose between:

* editing the decision table (table subview, **Table** button selected)
* editing macro definitions (text subview running in edit mode, **Macros** button selected)
* editing local variables (text subview running in edit mode, **Local Vars** button selected)
* editing the atomic function block’s comment (text subview running in edit mode, **Comment** button selected)
* inspecting the generated AIL code (text subview running in inspect mode, **Code** button selected)

* inspecting the selected interface item (text subview running in inspect mode, buttons deselected automatically)

If no interface item is selected, switching between the first five alternatives listed above is performed via related buttons.

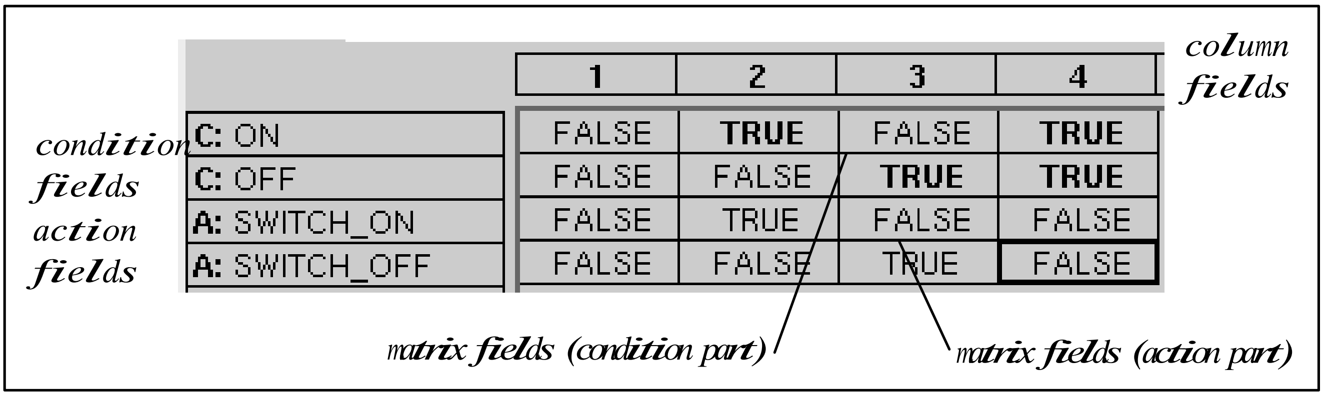


Figure 7–71 : *Structure of a decision table*

How to built a decision table

Figure 7–71 shows the general structure of a decision table. The conditions **C1:** and **C2:** concern the input variables called ON and OFF, both of type boolean. The matrix fields show all possible combinations in its condition part. As soon as a condition is added the condition part will be expanded automatically. The action part defines the actions which will be performed as soon as a certain condition is met.
Example: If the input variable ON is TRUE and the input variable OFF is false, the output variable SWITCH_ON is set to TRUE and the output variable SWITCH_OFF is set to FALSE.

There are two different ways to specify an action:

- you type the complete action in the action field and mark the desired condition with an asterisk in the matrix field or
- you specify the action incompletely, e.g. writing down the identifier of a variable that should be newly assigned. Inside the matrix fields the user has to specify either the value, an AIL expression to compute the value or a macro name.

Note that the maximum number of conditions and actions is 8.

*The labels **C1:** to **C8:** and **A1:** to **A8:** are automatically generated.*

	1	2	3	OTHERS
C: INPUT = 10	**TRUE**	FALSE	FALSE	–
C: INPUT > 10	FALSE	**TRUE**	FALSE	–
C: INPUT < 10	FALSE	FALSE	**TRUE**	–
A: OUTPUT_1	C_1			
A: OUTPUT_1		C_2		
A: OUTPUT_1			C_3	
A: OUTPUT_2 := INPUT + 1	×			×
A: OUTPUT_2		123		

Figure 7–72 : *A decision table showing three different ways to fill the action fields.*

Creating a decision table

- Create an atomic function block.
- Select an atomic FB with the left mouse button.
- Press the right mouse button and select **edit –> decision table.**
- Enlarge the decision table editor window.
- Press the right mouse button and select **add condition**.
- Type the condition into the text field.
- Press the **Accept** button. The condition and the related true/false columns appear in the decision table layout.
- Repeat the previous steps until you covered all conditions.
- *Note that there are different ways to fill the action fields.*
- Press the right mouse button and select **add action**.
- Type the desired action in the text field. (see Figure 7–72, row **A4:**)
- Select the action field in the matrix with the left mouse button.
- *to be continued ....*

- Press the right mouse button and select **toggle** from the pop–up menu. An asterisk appears in the selected row.
- OR
- Press the right mouse button and select **add action**.
- Type the name of an output in the text field.
- Select the action field in the matrix with the left mouse button.
- Press the right mouse button and select **edit** from the pop–up menu. Type an AIL expression, a local variable, a number or a macro into the text field. (see Figure 7–72, row **A1:,** row **A5:**)
- Press the **Accept** button.
- Press the **Apply** button to save the decision table.
- Select **Quit** from the window pop–up menu to close the decision table editor window.

Conditions and actions may be edited using macros or local variables in order to save space on the screen.

Note that you can use local variables and macros in the condition fields too.

Because the space in the different fields is limited, you can define local variables and macro definitions. The use of local variables and macros improves the readability of the decision table.

```
-- local variables

VAR_1, VAR_2, VAR_3 : UNSIGNED_INTEGER := 0;

VAR: UNSIGNED_INTEGER range 1..20 := INPUT;
```

Figure 7–73 : *Local variables definition in a decision table*

Editing Local Variables

- Press the **Local Vars** button.
- Type the definitions in the window.

Note that you have to follow the Ada syntax for the definition of local variables.

- Press the **Apply** button.
- Press the **Table** button to go back to the default decision table layout.

Editing macros

- Press the **Macros** button.
- Type the macro name, an equals sign and the macro expression in the window.

*Do **not** add a semi-colon at the end of the macro expression.*

- Press the **Apply** button.
- Press the **Table** button to go back to the default decision table layout.

```
– – macro definitions

C_1 = ( ( INPUT + INPUT ) / 5 ) + 1 + VAR_1 + VAR_2 + VAR_3 + NEW_VAR * 10

C_2 = ( ( INPUT + INPUT ) / 12 ) + 2

C_3 = ( ( INPUT * INPUT ) / 2 ) + 3
```

Figure 7–74 *: Macro definitions in a decision table*

Improve the table readability

If you add a new condition, the editor completes the decision table by automatically adding all the new condition combinations to the matrix.
The matrix can be minimized in two ways: you can put an entire column to the *others column* or you dismiss only one element in a column. (see the example in Figure 7–75)

Collect entire columns to the *other columns*

- Move the mouse pointer to the column number field and select it with the left mouse button.
- Press the right mouse button and select **remove** from the pop–up menu. The selected column disappears, a new column *others* appears in the matrix field if there is not already one.

There is a way to group columns which differ only in one condition which will not be used.

Group two columns to one

- Select a field in the condition part of the matrix with the left mouse button.
- Press the right mouse button and select **no matter** from the pop–up menu. The TRUE/FALSE is replaced by a # sign.

The decision table can be set to its original layout at any time, select the *others* or # field and perform the **expand** command.

	1		
c1: in1	**true**	**others**	minimized by others column
c2: in2	**true**		
a1: out1	true	false	incomplete AIL statement

	1	2	3
c1: in1	#	**false**	**true**
c2: in2	**false**	**true**	**true**
a1: out1 := true;			*
a2: out1 := false;	*	*	

Figure 7–75 : *Logical AND gate implemented by decision table (2 examples)*

*Select the **reset** button and you go back to the last saved version.*

7.4.7.5 Icon Editor

7.4.7.5.1 Basics

The Icon Editor runs in a separate MDE window. It may be activated from within the Composite Editor. The bitmap based black and white Icon Editor allows to inspect and edit the icons of composite and atomic function blocks.

Easy to understand iconized buttons allow the user to choose the kind of graphic operation he wants to execute. He may choose from:

* setting a single pixel
* drawing a line
* drawing a rectangle
* drawing a circle
* placing text into the icon
* selecting and moving a part of the icon’s bitmap
* selecting and copying a part of the icon’s bitmap

The editing operations are performed inside the icon edit subview via mouse. The edit subview shows a magnified copy of the original icon. It provides vertical and horizontal scrolling.

The user can specify the colour for setting single pixels, drawing lines, rectangles, circles and placing text. He may choose between black and white. The icon edit subview provides two modes for drawing rectangles and circles allowing to draw the border of the object or to draw a filled representation.

Optionally the user can decide to display a grid raster inside the edit subview. It is possible to clear the entire icon (set all pixels to white) and to invert the icon.

The current colour (black or white) and mode (border or fill) are shown in the Icon Editor’s status subview. Additionally the status subview shows the current text specified for the text placement operation.

Another non–interactive subview shows the icon in its original size.

User Interface

Figure 7–76 shows the user interface of the Icon Editor. It provides an icon edit subview. A bar of iconized buttons allows the user to select the current graphic operation. A non–interactive view displays the edited icon in its original size. A number of buttons and a text input field are used to show the current status.

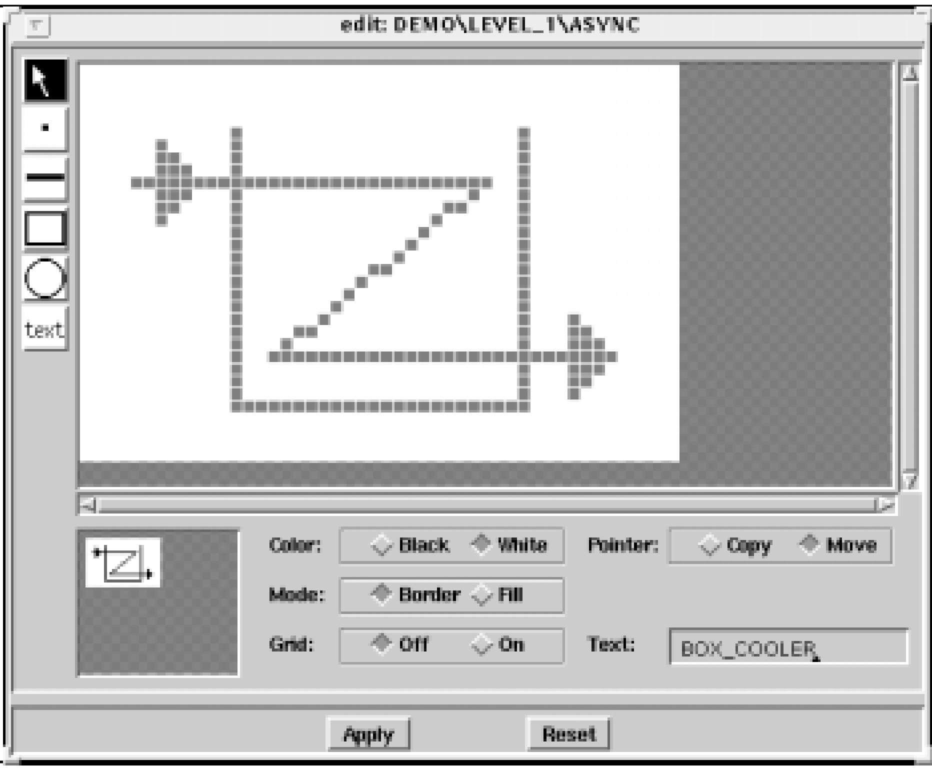


Figure 7–76 : *Icon Editor user interface*

7.4.7.5.2 Components

The Label

The label of the Icon Editor shows the pathname of the function block.

The Edit Subview

The edit subview is the place where icon editing actually is performed. It shows a magnified view of the edited icon. In general, drawing is performed by pressing and releasing the left or middle mouse button.

The Tool Buttons

By activating the different tool buttons, you specify what kind of object you want draw inside the edit subview. You can choose from pixel, line, rectangle, circle and text.

If the pointer button is activated, you can move resp. copy parts of the edited icon. To do so, press and hold down the left mouse button. A flashing rectangular outline appears on the screen. Move the cursor around and designate the area of the icon you want to move or copy, then release the button. Press and hold down the left mouse button again while moving the cursor around. The selected icon part appears highlighted following the cursor. Move it to the desired position and release the mouse button.

The Icon Subwindow

The icon subwindow shows the edited icon in its original size.

The Status Area

The status area shows the Icon Editor’s current status.

Color indicates the color that is used to display the objects you create inside the edit subview. The color can be set to *black* or *white*.

Mode indicates the mode that is used to display rectangles and circles. You can choose between *border* and *fill*.

Grid indicates whether inside the edit subview a grid raster should be displayed or not.

Text indicates the text you can create when the text tool button is activated.

Color, mode and grid can be set by activating the corresponding button. The text input field allows to type in the text to be created.

Working with the Icon Editor

You open the icon editor on a selected composite or atomic function block inside the Composite Editor. Initially the blocks are labeled with their name.

Create an icon and choose the menu command **Apply**. Note that on executing **Apply**, the selected function block inside the composite editor becomes labeled with the specified icon.

To remove the icon from a function block, select the function block inside the Composite Editor and execute the menu command **icon–>cut**. The icon disappears from the function block, it is labeled with its name again.

7.4.7.6 Tree Browser

The Tree Browser tool can be called within the Database Browser and on any level of the model hierarchy.

Note : Since the collection of the data required for the Tree Browser display, especially for a large model, may take quite some time, the user is prompted for confirmation before the Tree Browser is actually started.

After selection of a function block the user can jump via edit/inspect commands to the related hierarchy level.

The tree browser provides the same documentation generation function (command **print**) as the composite editor resp. inspector (see the related chapters).

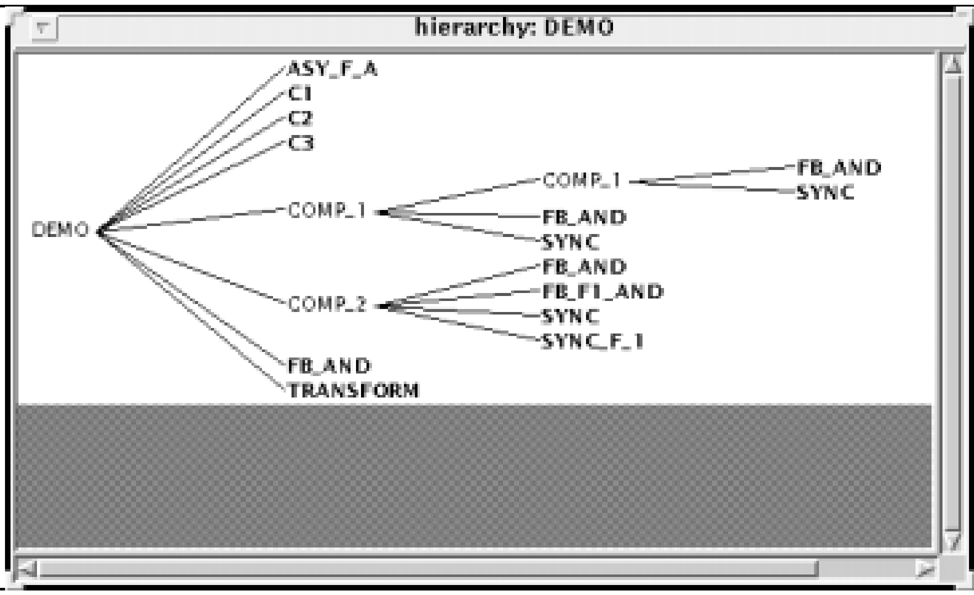


Figure 7–77 : *The Tree Browser user interface*

Navigate in the model tree structure

- Select **goto** in the Composite Editor's basic menu. The tree browser window (see Figure 7–77) appears.
- Scroll to the desired part of the model.

*Note that atomic function blocks and parameter blocks are written in **bold** style.*

- Click on the desired item. The item is highlighted.

Perform the next two steps if the selected item is a composite FB.

- Press the right mouse button and select **edit–>implementation** from the pop–up menu. The Composite Editor switches its currrent level and shows the selected composite FB's inside view in overview mode.
- Select **edit** from pop–up menu.

OR

Perform the next step if the selected item is an atomic FB.

- Press the right mouse button and select **edit–>AIL** or **edit–>decision table** from the pop–up menu. An atomic editor window appears.
- Select **Quit** from the window menu of the tree browser window.

7.4.7.7 Documentation Generation Function

Note that there are several ways to get model print–outs.

The Documentation Generation Function runs in a separate MDE window. It may **only** be activated from within the Database Browser.
Refer to section 7.4.7.1.41 where is described how to get a listing while you are inside a model. Note that every editor provides a separate printing function.

The Documentation Generation Function provides a graphic subview showing a vertically and horizontally scrollable tree view that allows to select the desired parts of the model by clicking the mouse on the item.

The user can choose between a print on a laser printer or the creation of a text document which can be handled by the CGS SDE text system Interleaf.

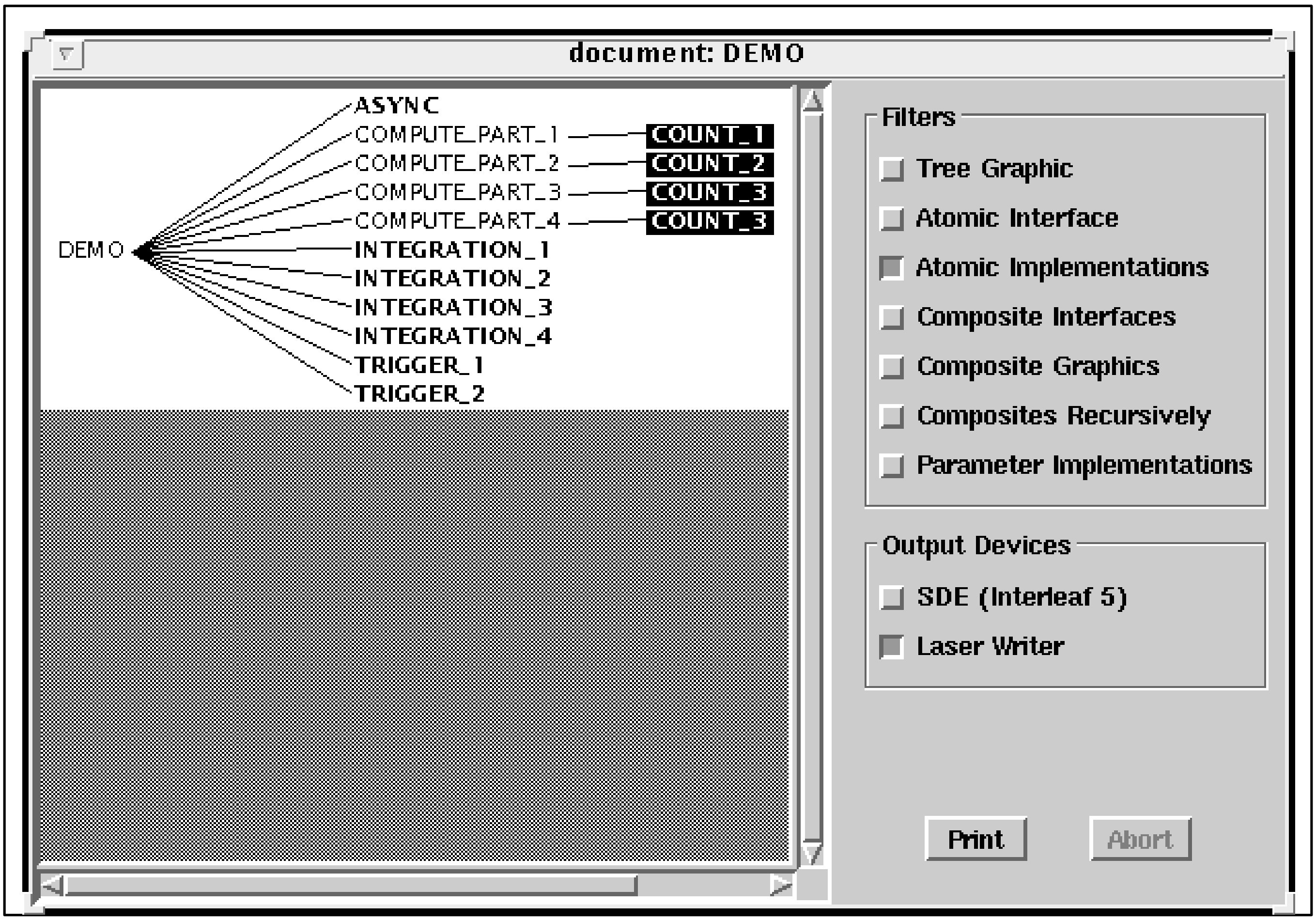


Figure 7–78 : *Printing the contents of the atomic FBs COUNT_1 to COUNT_4*

Printing model documentation

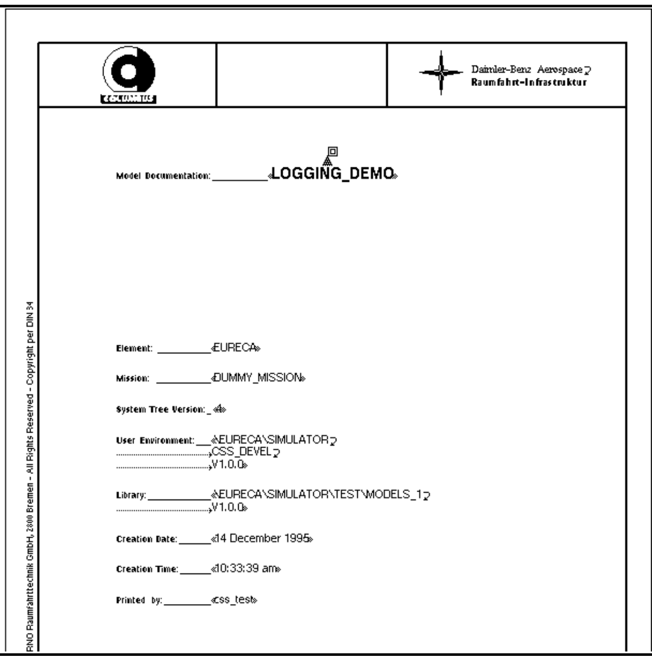
- Open the Database Browser window.
- Select the model by clicking on its name with the left mouse button.
- Press and hold down the right mouse button.
- Move the mouse pointer to the **print** command and release the mouse button. The documentation generation window (see Figure 7–78) appears.
- Use the scroll bars to display the desired part of the model.

*Note that the **Tree Graphic** Filter is the only item which needs no selection in the displayed model tree structure (tree subview).*

- Click on the desired function blocks in the model tree structure (tree subview). The names become highlighted.
- Select the contents of the printout in the **Filters** list.
- Select the **Output Device**. Click on either the **Laser Writer** button or (to get a text document) the **SDE** button.
- Press the **Print** button.
- Select a printer from the pop–up list of available printers.

*Note that the contents of the printer list depends on the installation environment. If there is only one printer available a field **default** appears.*

To create the model description as an Interleaf document, follow the steps in the procedure. Select **SDE** as the output device. Automatically a book with the models name will be created in your desktop directory.


Daimler-Benz Aerospace
Raumfahrt-Infrastruktur

Model Documentation: LOGGING_DEMO

Element: EURECA
Mission: DUMMY_MISSION
System Tree Version:
User Environment: \EURECA\SIMULATOR
CSS_DEVEL
V1.0.0
Library: \EURECA\SIMULATOR\TEST\MODELS_1
V1.0.0
Creation Date: 14 December 1995
Creation Time: 10:33:39 am
Printed by: css_test

RNO Raumfahrttechnik GmbH, 2800 Bremen – All Rights Reserved – Copyright per DIN 34

Figure 7–79 : *The cover page of the model document*

Modifying the default model documentation layout

- Invoke the documentation tool.
- Open the book generated by CSS (the name of the book is equal to the model name).
- Open and change the catalogs Definition and/or Main_Catalog.
- Save your changes.

The modification of the model document layout for all CSS users is a task which can be performed only by a priviledged user (e.g. system administration). The standard book layout is located in the $CSS_HOME/config/ileaf directory. Copy the book CSS.boo to your desktop, make the required changes and copy it back. From now on all CSS users will get the new layout in the model documentation.

Note that pictures taken from the model are not visible in the Interleaf document. Nevertheless they appear as soon as the page is printed.

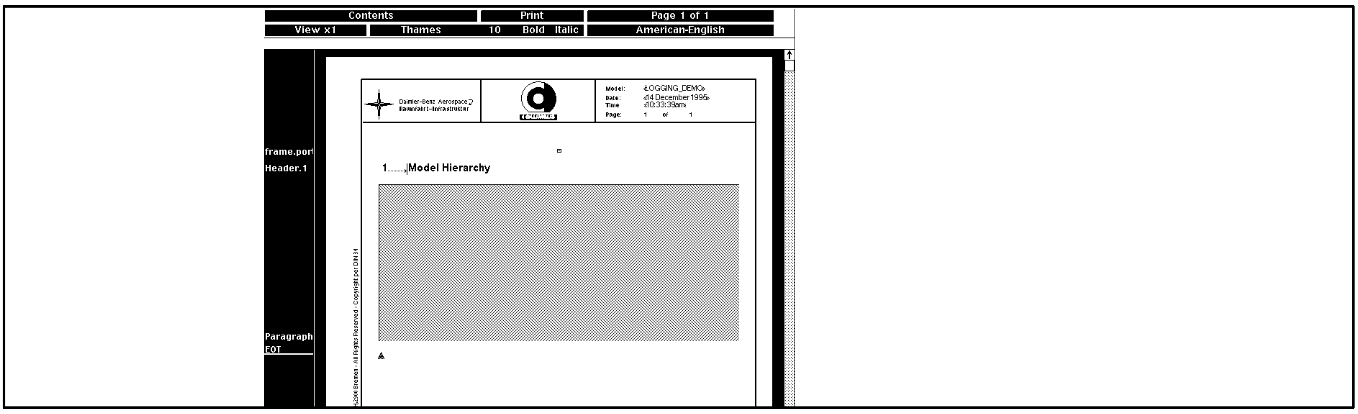


Figure 7–80 : *This grey rectangle shows the model hierarchy after printing*

7.4.7.8 Onboard References Adaptation Function

7.4.7.8.1 Basics

Each interface item of a database based simulation model may reference an onboard end item contained in the database, i.e. it may keep an onboard reference which is actually the database pathname of an onboard item. When a database based simulation model is loaded, its onboard references are checked on validity (accessability of the referenced onboard items in the current CCU scope and type compatibility, if appropriate). All onboard references that have become undefined and are not adapted, will be reset automatically, i.e. the onboard references are cleared and therefore lost.

An onboard reference is said to be undefined, if an onboard item with the given pathname cannot be accessed (is not visible) in the current CCU scope. There may be a large number of undefined onboard references if a model is loaded under the wrong CCU by mistake. In such a case, the windows opened on the model should be closed without saving the model, and then the model can be reloaded under the correct CCU.

However, it may also be the case that individual onboard items have been renamed, removed and substituted by other onboard items, or that entire subtrees have been moved to another location in the database. In order to avoid the need for setting once again all these onboard references one by one using the Composite Editor resp. Composite Interface Editor in conjunction with the Database Browser, CSS provides the Onboard References Adaptation Function which facilitates the task of adapting the simulation model's onboard references.

The Onboard References Adaptation Function runs in a separate MDE dialog window. To activate it, make sure that the simulation model is not loaded, i.e. there may be no open window on the model. Then select the simulation model in the item subview of the Database Browser and choose one of the commands **Edit–>Implementation** or **Edit–>Interface** from the pop–up menu. If the model contains undefined onboard references, the dialog window of the Onboard References Adaptation Function appears on the screen allowing to adapt these onboard references before the resp. editor window is opened; otherwise the Composite Editor resp. Composite Interface Editor window is opened directly.

User Interface

Figure 7–81 shows the user interface of the Onboard References Adaptation Function. It provides two scrolling list subviews showing the undefined and adapted onboard references. Two text input fields and a number of buttons allow adapting of onboard references. A read only text subview is used to provide textual feedback to the user.

Working with the Onboard References Adaptation Function

The upper scrolling list subview labeled **Undefined Onboard References** provides the list of yet undefined onboard references, the lower one labeled **Adapted Onboard References** shows the list of meanwhile adapted onboard references. Both subviews allow multiple selection. Below each subview, if exactly one of the listed onboard references is selected, the name of the referencing model interface item (i.e. top level I/O) is given.

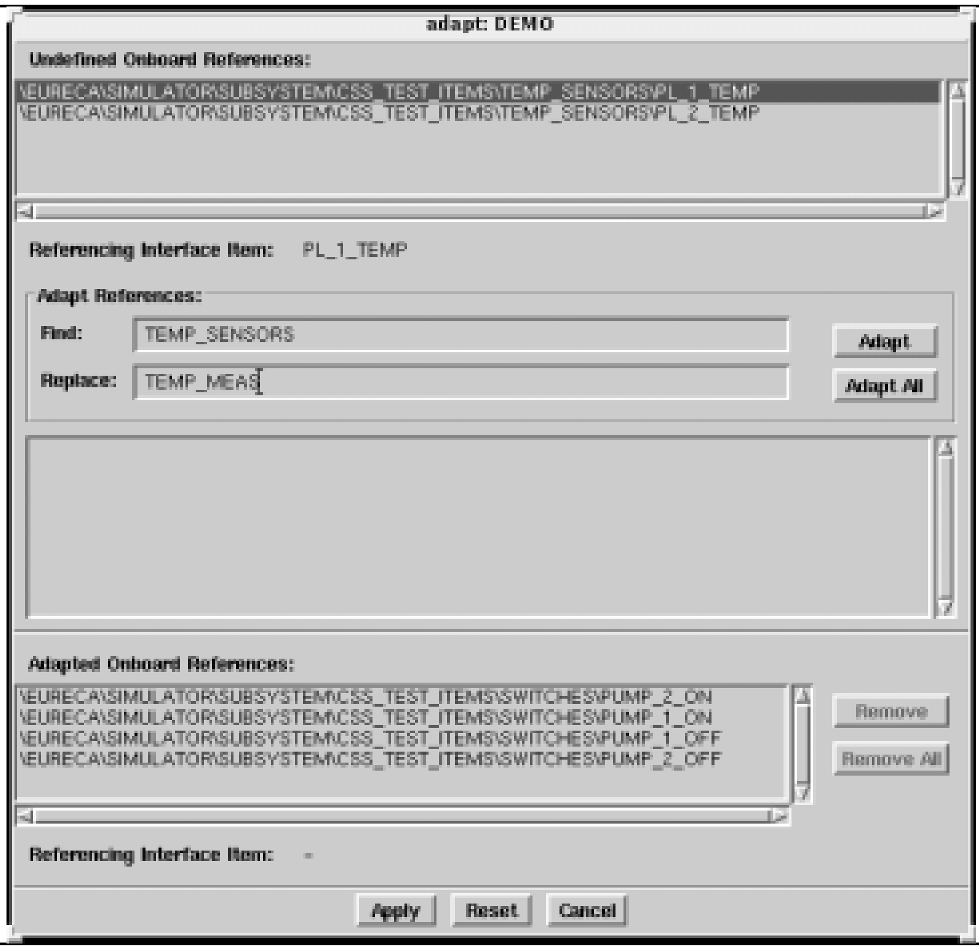


Figure 7–81 : *The Onboard References Adaptation window*

Adaptation is performed using the two text input fields. In an undefined onboard reference, the first occurence of the characters typed in the text input field labeled **Find** is searched for, and if found, is replaced with the characters typed in the text input field labeled **Replace**. Then an attempt to set the onboard reference to this new generated pathname is made, and if successful, the undefined onboard reference is removed from the upper scrolling list subview and the new onboard reference appears in the lower scrolling list subview. An attempt that failed, e.g. because an end item could not be located in the current CCU scope or the item found is already referenced by another model interface item, is indicated in the message subview (the read only text subview).

Pressing the **Adapt All** button applies adaptation to all undefined onboard references provided in the upper scrolling list subview while pressing the **Adapt** button restricts adaptation to the undefined onboard references currently selected in this subview.

The **Remove** resp. **Remove All** buttons allow to remove the selected resp. all already adapted onboard references from the lower scrolling list subview.

Pressing the **Apply** button completes the adaptation process by setting all adapted onboard references shown in the lower scrolling list subview. Pressing the **Cancel** button aborts the adaptation process without adapting any onboard reference. The **Reset** button allows to discard any adaptations performed so far and to restart the adaptation process from the beginning.

7.4.7.9 The Simulation Table Editor Window

7.4.7.9.1 Creating a Simulation Table

The Simulation Table Editor runs in a separate window. It may be activated from within the DBB window. The tool allows to define the monitoring, logging or tracing elements needed to observe the behavior of dynamical values during simulation.

Creating a Simulation Table

- Open the Database Browser window. (see Figure 7–82)
- Select a model by clicking on its name with the left mouse button. The model name becomes highlighted.
- Press the **Simulation Tables** button with the left mouse button.
- Move the mouse pointer into the Info Subview II (lower right field) of the DBB.
- Press and hold the right mouse button. Then move the mouse pointer to the **add table** entry in the pop–up menu. Release the mouse button. An input window appears.
- Type a new table name (a default name is already given) in the input field.
- Press the **Apply** button. The table name appears in a list in Info Subview II.

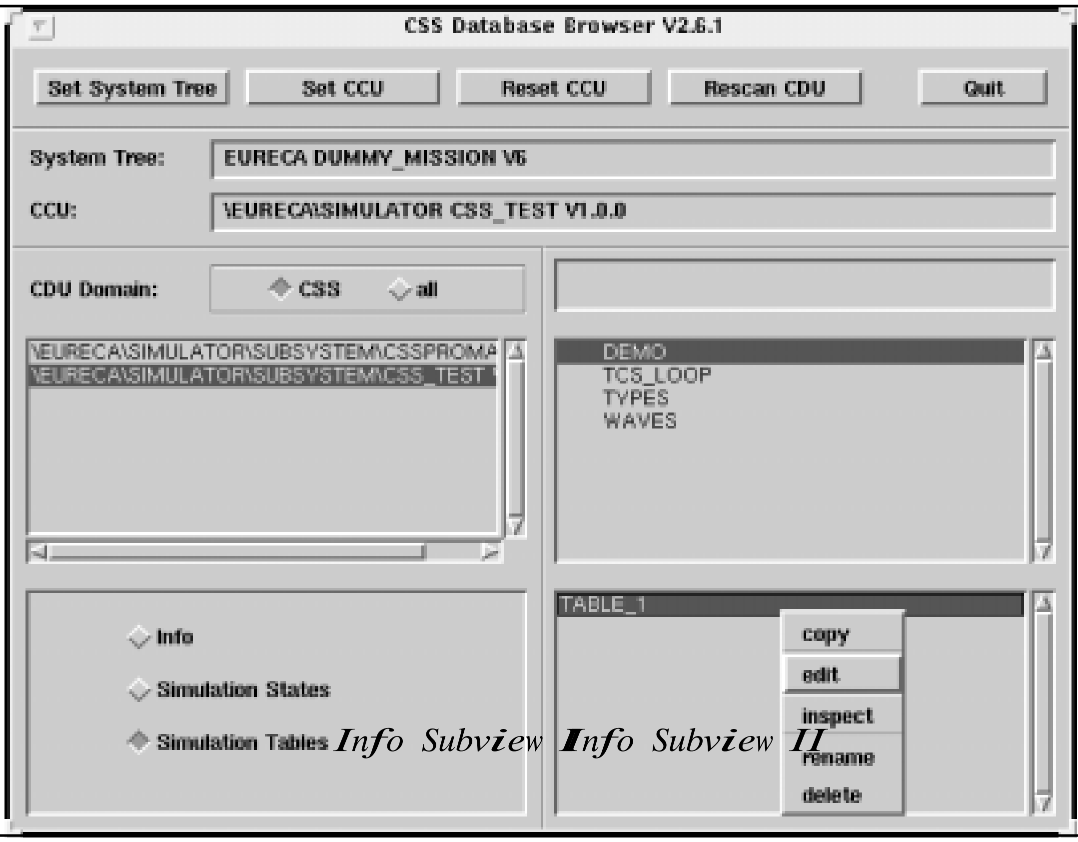


Figure 7–82 : *The Simulation table TABLE_1 is selected in the DBB window for editing.*

The Simulation Table Editor is a tool that allows to define model items for monitoring , logging and tracing.

* Monitoring means to observe online dynamic changes of the model parameters during simulation execution.
* Logging means to store the change of dynamic values for offline evaluation.
* Tracing means to observe the moment of activation (execution) of functions blocks on the screen, or to record the moment of activation in a log file.

The simulation table editor window in Figure 7–83 shows on the upper left side the name of the currently edited table with the table status in parantheses. The default table status is ’unchanged’, while editing the table the table status changes to ’changed’.

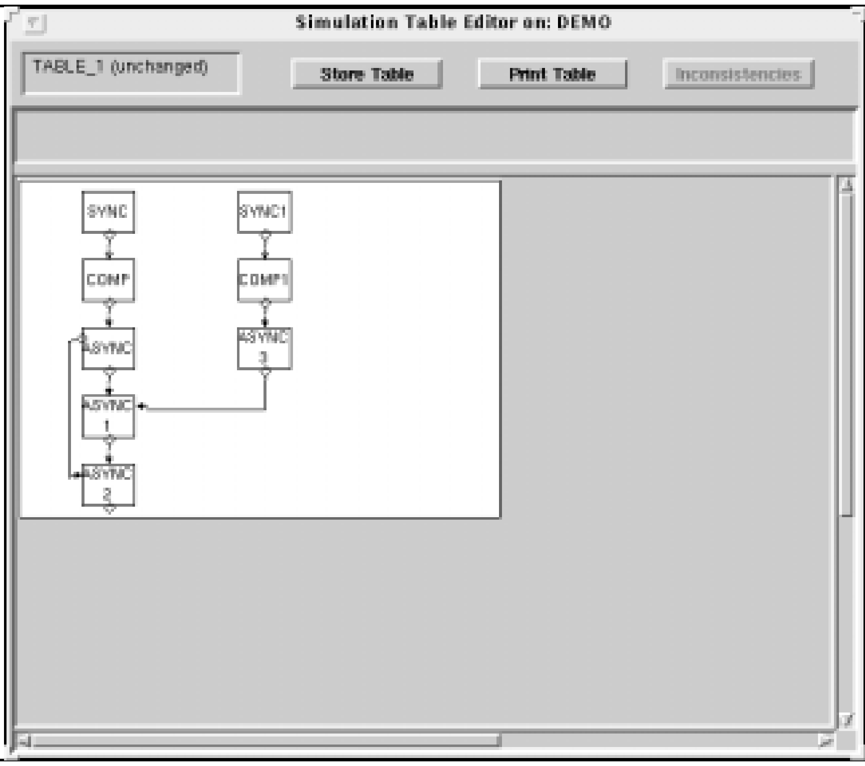


Figure 7–83 : *The Simulation Table editor window (overview mode)*

The three buttons in the upper field have the following functions:

* the **Store Table** button
  stores the monitoring table. If you wish to create a new table instead of changing the old one change the name in the appearing input window.
* the **Print Table** button
  prints the contents of the simulation table as ASCII listing, e. g. the complete item name, the activation status, the selected type of monitoring element, logging and tracing items.
* the **Inconsistencies** button
  is normally disabled. If you load a simulation table which does not fit to the currently selected model this button becomes active. A list of inconsistencies between the model and the selected table will be displayed.

Note that it is possible to have more than one open simulation table during model execution on the screen. In models with several levels the user may observe the changes on different levels in parallel.

- *Due to the fact that there is no* switch on/off *feature for tracing (the tracing starts as soon as a table containing tracing items is loaded and the model execution is started) the user should split the functions, i.e. the user may decide to put all monitoring and logging items in one table or split them into several tables, but tracing items should be in a separate table in any case.*

7.4.7.9.2 Creating monitoring items in the simulation table

Prerequisite for the monitoring of dynamic values is to assign a value monitoring element to the desired I/O item.

In the Simulation Table the graphical monitoring elements appear on the background of a graphical model structure, just the same representation as for MDE.

In order to enter new monitoring definitions, the user can navigate through the model structure and select the desired item in the model’s graphical representation.

The provided value representation types for graphical monitoring are:

* literal representation
* gauge representation (horizontal bar, vertical bar and dial)
* curve representation
* discrete state representation

- *Editing a table for monitoring means to assign a literal, gauge, curve or discrete state representation element to the desired inputs or outputs.*

Editing monitoring elements in a Simulation Table

- Open the Database Browser window.
- Select the model by clicking on its name.
- Press the **Simulation Tables** button.
- Select the Simulation Table by clicking on its name in the list.
- Press and hold the right mouse button. Select **edit** from the pop–up menu. The simulation table editor window appears (in overview mode).
- If the editor window appears in overview mode, move the mouse pointer into the graphic subview and select **inspect** from the pop–up menu.
- Click on the desired I/O item and select **monitor–>literally** (or any other type of monitoring element) from the pop–up menu. An input window for monitoring parameters pops up. (see Figure 7–84)
- Click in the **on change** box in the monitoring parameter window.

*It is recommended to use the **on change** monitoring option if possible.*

OR

- Click in the **cyclic** box in the monitoring parameter window.
- Type a number between 1 .. n in the Cycle field. The value will be monitored every nth frame.
- Press the **Apply** button. A monitoring element is displayed next to the selected I/O item.
- Click on the monitoring element and resize or move it conveniently.
- Repeat the preceding four steps until all desired items are monitored.
- Press the **Store Table** button. An input window appears.
- Change the table name in the input field, if the table shall be stored under a new name.
- Press the **Apply** button.
- Choose **Quit** from the window pop–up menu. The table editor window disappears.

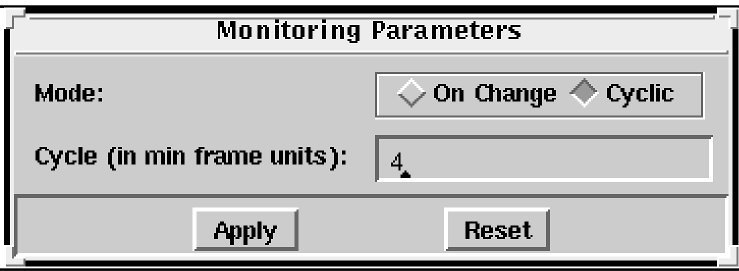


Figure 7–84 : *The Monitoring Parameters window*

On change means that the monitoring value will be displayed each time the value changes. During simulation run the monitoring table display update rate is once a second.
Note that for values changing faster than once a second you can't observe each value during continuous simulation. Choose single step simulation mode for proper monitoring.

If you select a **cyclic** monitoring note that the minimum cycle time is not limited but the same display restrictions appear.

In the preceding procedure it is recommended to use the **on change** monitoring. This is valid for parameters which do not change too often. If a parameter changes very rapidly it can be advantageous to select the **cyclic** monitoring.

In the preceding procedure only one type of monitoring element is mentioned. Literally monitoring means that the value is displayed as a string.
Depending on the type of value you want to monitor you get a list of possible graphical elements.

The provided value representation types for graphical monitoring are shown in Figure 7–85:

* literal representation
  The value is displayed as a textual string,
  this representation is possible for all value types.
* gauge representation
  The value is displayed either as horizontal or vertical bar graph or in a dial display,
  this representation is possible for all numerical types incl. duration values.
* discrete state representation
  Discrete state representation is applicable for items with a small set of alternative values, the values are displayed as a list, the current value is shown by setting a marker, this representation is available for boolean and state_code values.
* curve representation
  The display shows the temporal behaviour of a value. At defined steps the value is shown, not overwriting the preceding value. This representation is applicable to numerical values.

Note that monitoring can only be performed for scalar items (not allowed for vector, matrix etc data types). Use the snapshot feature during simulation execution instead.

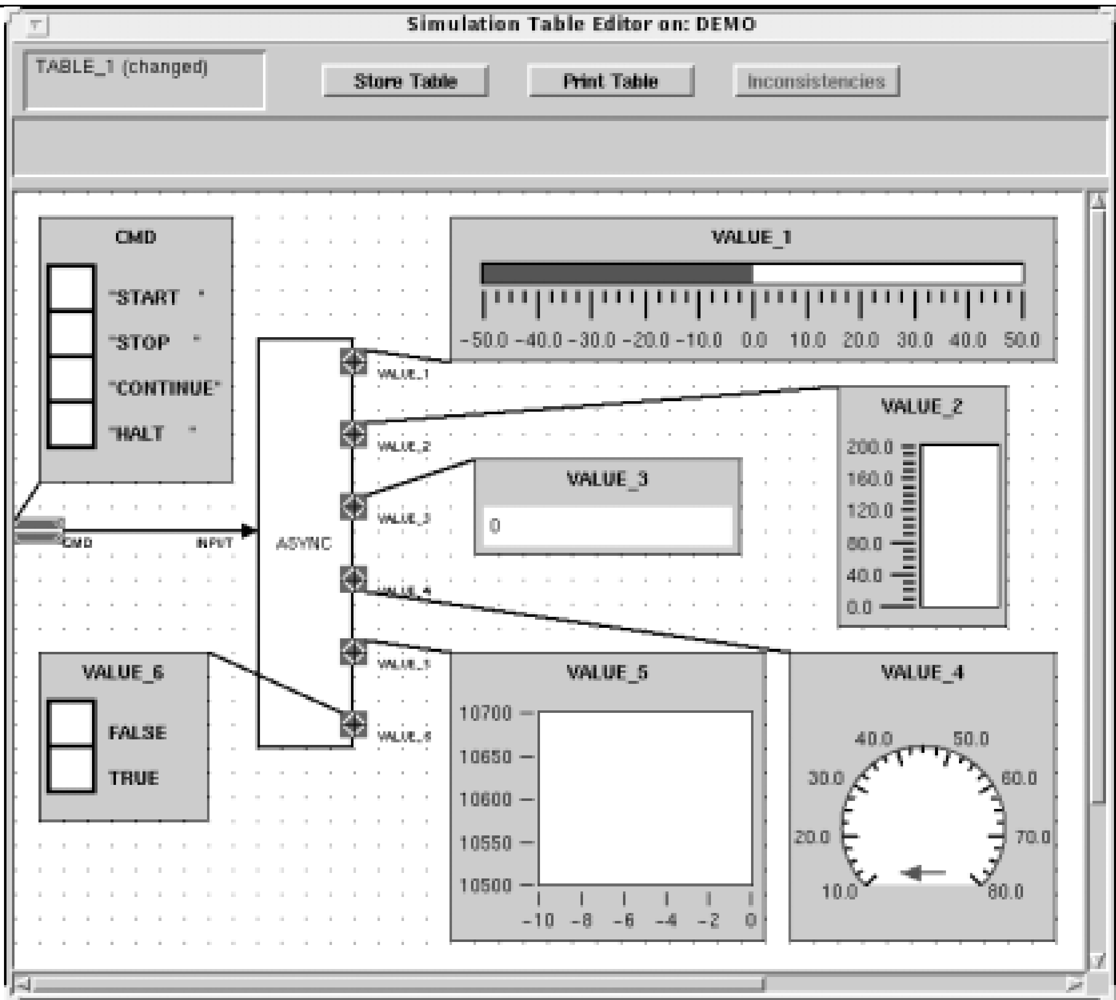


Figure 7–85 : *The graphical representation elements*

Gauge elements have a default scaling which can be changed easily.

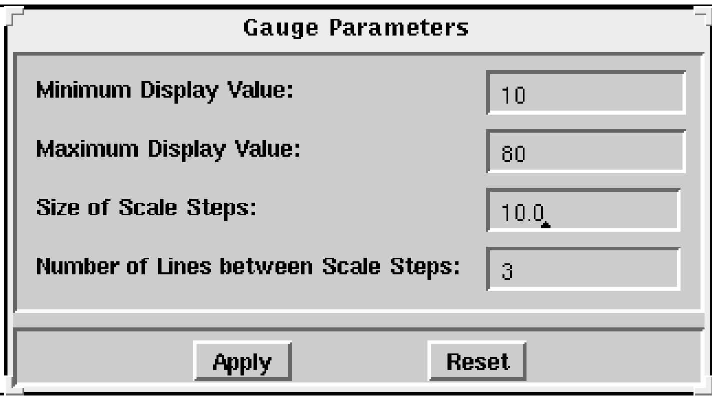


Figure 7–86 : *The gauge element scaling parameters*

Minimum and maximum display value define the range of the scale. The size of scale steps determines the distance between the numbering and the precision of scale lettering. If you write the number with a decimal point and digits behind the decimal point the lettering will be in this format.
Example: display a value between 0 and 10. Size of scale steps shall be two. The scale lettering is 0,2,4,6,8,10. If you select a scale steps size of 5, you see the lettering 0,5,10 on your scale.

The number of lines between the scale steps determines the subdivision of the scale.

Changing the scaling of gauge monitoring elements

- Click on the monitoring elements with the left or middle mouse button. The monitoring elements appear selected.
- Press and hold the right mouse button. Select **parameters** from the pop–up menu.
- The Gauge Parameters window (see Figure 7–86) appears.
- Change the parameters accordingly.
- Press the **Apply** button.
- Check the display and repeat the preceding steps if necessary.

You can delete monitoring elements either by selection of the element and **remove** from the menu or by clicking on the I/O–item and **remove definitions.** The latter command removes all monitoring elements connected to the I/O item.

The other commands available within the simulation table editor window do not differ from the commands in the MDE windows described in previous chapters.

7.4.7.9.3 Restrictions on monitoring

Monitoring elements can be assigned to all atomic and top–level inputs/outputs.
A different point are the borders of a composite function block. Inputs and outputs can be monitored either within the composite function block or outside, but not on both levels.

As Figure Figure 7–87 shows it is not possible to assign a monitoring element to the input on the left side of the composite function block. The inputs to a composite function block can be monitored only on the next lower level (inside view).

On the other hand it is not possible to monitor the output on the inner right side of the composite (see Figure Figure 7–87 – inside view). Outputs to the next higher level can be monitored on the higher level only (the outside view shows the monitoring element assigned to the output).

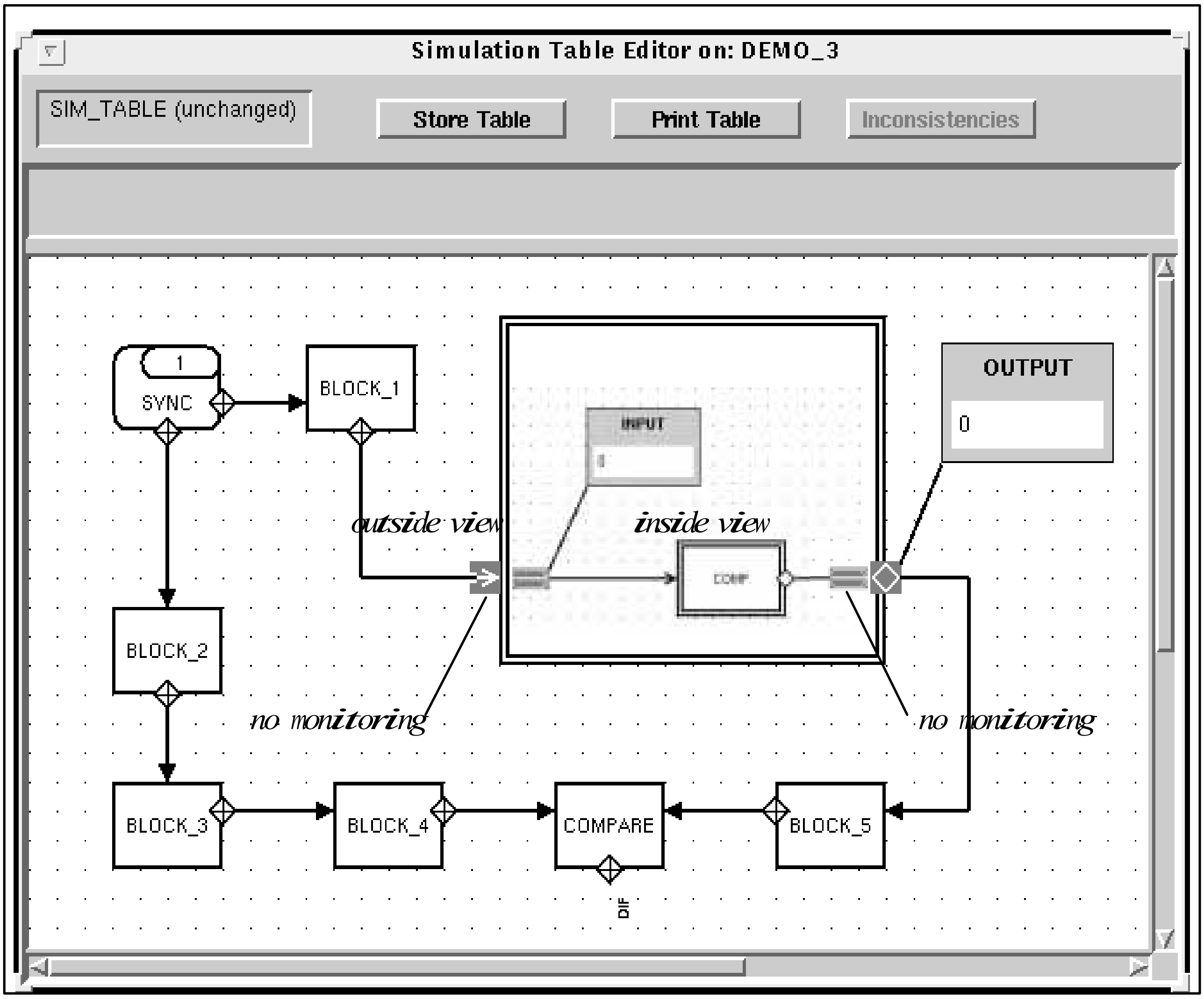


Figure 7–87 : *Monitoring I/O items*

I/O items at the border of a composite function block monitored on a level which is currently not visible are marked with a grey underlay.

7.4.7.9.4 Creating logging items in the simulation table

An interface item can be logged **on change** (i.e. the value is written each time the value changes) or **cyclically** each minframe. The items marked for logging are displayed in the simulation table editor window with a grey underlay.

The logging results are stored in *archive files* (ASCII text format) for post simulation evaluation. Each entry listed comprises the logged item's RID, the variable type, a value and the time (local time and simulated mission time) the value was logged.

In addition, it is possible to store the results in *data set files* (binary format) which can be processed by the Test Evaluation S/W (TEV); for more information on data set processing refer to the TEV user manual.

Editing logging elements in a Simulation Table

- Open the Database Browser window.
- Select the model by clicking on its name.
- Press the **Simulation Tables** button.
- Select the Simulation Table by clicking on its name in the list.
- Press and hold the right mouse button. Select **edit** from the pop–up menu. The simulation table editor window appears (in overview mode).
- If the editor window appears in overview mode, move the mouse pointer into the graphic subview and select **inspect** from the pop–up menu.
- Click on the desired I/O item and select **log–>on change to archive** resp. **log–>on change to archive and data set** from the pop–up menu. The I/O item is shaded with grey. (see Figure 7–88)

  OR
- Click on the desired I/O item and select **log–>cyclically to archive** resp. **log–>cyclically to archive and data set** from the pop–up menu. The I/O item is shaded with grey.

Note that **log–>cyclically** *means that the values are recorded every min frame.*
The recommendation is to use the ***on change*** *option.*

- Press the **Store Table** button. An input window appears.
- Change the table name in the input field, if the table shall be stored under a new name.
- Press the **Apply** button.
- Choose **Quit** from the window pop–up menu. The table editor window disappears.

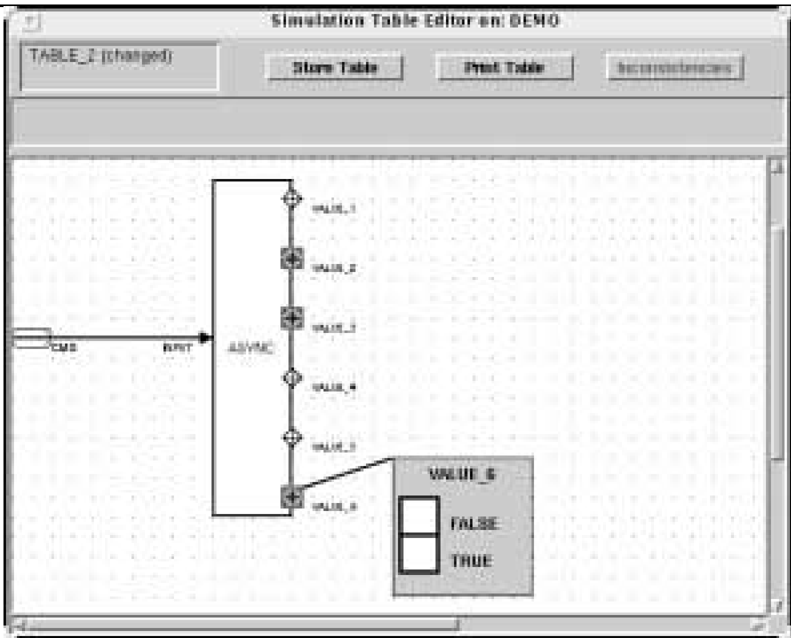


Figure 7–88 : *The outputs VALUE_2, VALUE_3 are logged*

The files containing the logging results (i.e. the *archive files* and optionally the *data set info files* and *data set files*) will be stored as soon as the session owner finishes the simulation session (i.e. the simulator kernel

is stopped) or a user explicitely presses the button **Logging: Store Results** in the Simulation Controller; this is possible repeatedly while the simulation is running. Each time the logging results are stored, new files are created comprising the data accumulated since the last storing or since the start of the simulation; a serial number is appended to the file names.

A *data set info file* (ASCII text format) provides information about the associated *data set file(s)*, i.e. for each associated *data set file* its pathname, the timeframe (start and end) covered and some statistics concerning the interface items logged, i.e. for each item pathname and RID and the number of values contained. The *data set files* contain the logged values in binary format.

Each time a simulator kernel is started, the system generates at least the *log file*. It contains the pathname of the mapping table which maps the pathnames of interface items to their runtime identifiers (RIDs) and the listing of all commands given during the simulation session.

The *log file* and the *archive files* are stored in the Unix file system in a directory which is determined by the environment variable CSS_LOG_DIR. The *log file* is marked with the extension .LOG_TXT. The *archive files* are marked with the extension .ARCH_TXT_#, the sharp sign indicates the serial number. In the same directory, the *data set info files* are stored. They are marked with the extension .DATA_SET_INFO_#. The sharp sign indicates the serial number, each file corresponds to the archive file with the same number. For each of these files the part of the name preceeding the extension matches the pattern <model name>_<user name>_<architecture>_DDMMYYYY_HHMM. First comes the simulation model's name (<model name>), then an underline followed by the name of the user (i.e. the name of the user account) who started the simulator kernel (<user name>). Next comes an underline and then the architecture of the host the simulator kernel is/was executing on (<architecture>) followed by another underline. This is followed by 8 characters denoting the date (DDMMYYYY, i.e. day, month and year), a further underline and 4 characters denoting the time (HHMM, i.e. hours and minutes) the simulator kernel was started.

The *data set files* are stored in the UNIX file system in a directory which is determined by the environment variable CSS_LOG_DATA_SET_DIR. Their names match the pattern <model name>_DDMMHHMM_##. First comes the simulation model's name (<model name>), then an underline followed by 4 characters denoting the date (DDMM, i.e. day and month) and 4 characters denoting the time (HHMM, i.e. hours and minutes) the logging results were stored. The serial number (first sharp sign) follows a further underline; it associates the *data set file* to the corresponding *data set info file* with the same number. In order to avoid too large files, a data set may be automatically partitioned furthermore over a subseries of *data set files*; the last character (second sharp sign) denotes the position in this subseries. Because the names of *data set files* are restricted to 20 characters, <model name> may miss some trailing characters of the actual model name.

If several users are connected to one simulation session (each of them with his/her own logging definitions) there will be only one table with logging results. It is the users task to identify their own logging values.

Warning:
Each time you start a model execution the system generates at least the log file (i.e. the file with the extension .LOG_TXT) automatically. It is not possible to suppress this feature. You have to delete this files from time to time or you get problems with your disk quota (i.e. the available space on your computer).

7.4.7.9.5 Restrictions on logging

Logging definitions are subject to the same restrictions as described in section 7.4.7.9.3. It is not possible to log an input connected to the outer border of a composite function block, the same goes for outputs connected to the inner side of a composite.

Interface items of the following types may not be logged to data sets: DURATION, TIME, COMPLEX, all VECTOR types and all MATRIX types.

7.4.7.9.6 Creating tracing items in the simulation table

There are two ways to get the tracing information.

Each time the function block is activated during simulation execution you get the current date, the SMT and the run time ID of the activated FB either into a separate window (**trace to screen**) or written to the log file (**trace to log**). The function blocks marked for tracing are displayed in the simulation table editor window with a grey underlay (see Figure 7–89).

Editing tracing elements in a Simulation Table

- Open the Database Browser window.
- Select the model by clicking on its name.
- Press the **Simulation Tables** button.
- Select the Simulation Table by clicking on its name in the list.
- Press and hold the right mouse button. Select **edit** from the pop–up menu. The simulation table editor window appears (in overview mode).
- If the editor window appears in overview mode, move the mouse pointer into the graphic subview and select **inspect** from the pop–up menu.
- Click on the desired function block and select **trace–>to log** from the pop–up menu. The function block is shaded with grey. (see Figure 7–89)

  OR
- Click on the desired function block item and select **trace–>to screen** from the pop–up menu. The function block is shaded with grey.

Note that composite function blocks can‘t be traced.

- Press the **Store Table** button. An input window appears.
- Change the table name in the input field, if the table shall be stored under a new name.
- Press the **Apply** button.
- Choose **Quit** from the window pop–up menu. The table editor window disappears.

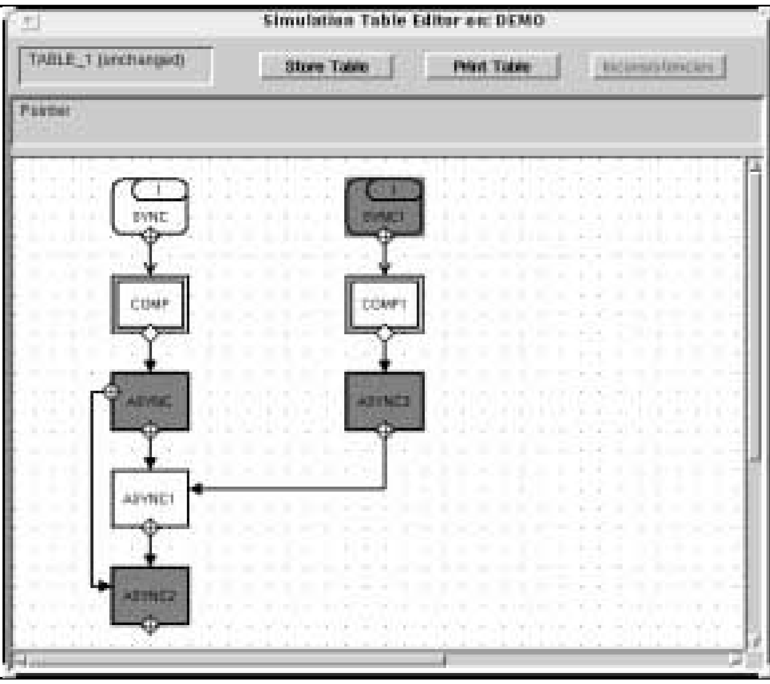


Figure 7–89 : *The function blocks marked for tracing*

7.4.7.9.7 Restrictions on tracing

If several users are connected to one simulation session (each of them with his/her own tracing definitions) there will be only one table containing the tracing informations.
The file containing the tracing results will be written as soon as the session owner finishes the simulation session (i.e. the simulator kernel is stopped).

Warning:
If you use the trace to screen option on a synchronous function block running with frame 1 (minframe) and in continuous mode this will block your workstation completely, even a remote login is not possible. The only way to get out of this situation is to re–boot the workstation.
Recommendation : Do NOT use this feature.

7.4.8 CSS Configuration Environment Variables

Several shell environment variables control the behaviour of CSS, usually to create additional execution tracing information for troubleshooting. Some may be set in an interactive shell, just before starting the standalone version of CSS (`$CSS_HOME/bin/common/CSS`), some have to be set in the user's `.cshrc` script resp. in a script called `$HOME/.user/css_cshrc` in order to be visible for CSS components started via remote shell or from I_MDB.

- CSS_DEFAULT_STATE_CODE_HANDLING (cshrc)
  set to TRUE to process also the default state code for discrete onboard data types when checking the compatibility to the connected model I/O. Otherwise, the default state code will be ignored.
- CSS_FILE_BASED (interactive)
  set to TRUE to start CSS without connecting to the MDB, working only with models stored in the file system.
- CSS_LOG_DIR (cshrc)
  Directory for storing all CSS logging and debugging output; default is : `$HOME/css/logger`
- CSS_LOG_DATA_SET_DIR (cshrc)
  Directory for data sets produced by the logger_be; default is : `$HOME/wd/tev/RESULTS/DATA_SET`
- CSS_MODELS (cshrc)
  Directory for MDB based model files; default is : `$HOME/models.lib`

Variables for synchronization purposes between the CSS kernel and CMAS:

- SET_KERNEL_STATUS_TIMEOUT ($CSS_HOME/user_env/cshrc)
  The value is expressed in milliseconds and defines the time the CSS kernel will wait for an answer from CMAS if the CSS status has changed (e.g. SIMULATION_RUNNING or SIMULATION_STOPPED). The default value is set to 5 seconds, i.e. if the variable is not defined a value of 5000 ms will be assumed. CMAS will need a certain amount of time to initialize internal and associated structures.
- CMAS_INITIALIZATION_TIMEOUT ($CSS_HOME/user_env/cshrc)
  The value is also expressed in milliseconds and defines the allowed time interval for CMAS initialization. The CMAS process is started by the CSS kernel and will initialize internal structures, connected hardware (e.g. MILBus boards) etc. The status of CMAS will be checked by the CSS kernel if a new statevector has to be loaded. In case the CMAS status is not correct and this time interval has expired a warning message will be produced and displayed via the CGS error services. The default value is set to 120 seconds.

Variables for additional debugging information to support troubleshooting:

- CSS_DBS_OPTIONS (cshrc)
  When set, the CSS DB Server will write additional debugging information to files at $CSS_LOG_DIR/db_server.message_log.‘hostname‘.$$
  Possible settings are : “–v<n>”, with n=0..9 indicating different levels of verbosity.
- CSS_LOGGING (cshrc)
  When set to TRUE, the CSS CTG Server will write additional debugging information to files at $CSS_LOG_DIR/ctg_server.message_log.$USER.$HOST.$$

7.4.9 File System Maintenance

During preparation and execution of a simulation, CSS saves information in several kinds of log and report files, which eventually ma y consume a significant amount of disk space. Occasionally, it may become necessary to clean up these files, either due to lack of disk space, or just because they are not needed any more.

- **`$HOME/CSS_SCOPE_CHECK.LOG`** :
  This file accumulates the reports from the CSS scope check window that appears when launched by I_MDB. The latest report will be found at the end, so this file should be frequently deleted to avoid excessive scrolling.
- **`$CSS_LOG_DIR`** (usually **`$HOME/css/logger`**) :
  This directory contains log and report files from several CSS components:
  - **`<model>_<architecture>_<date>_<time>.LOG_TXT`**
    `<model>_<architecture>_<date>_<time>.ARCH_TXT_*`
    `<model>_<architecture>_<date>_<time>.DATA_SET_INFO_*` :
    Contain logged data and archived information from simulation execution sessions. This would usually be important simulation results and should be archived somwhere else, or at least be examined carefully, before deletion !
  - **`<model>_<architecture>_<date>_<time>.COMPILATION_REPORT_TXT:`**
    Contain messages created by the CTG server during compilation of a model. These are just backup copies of the messages displayed in the MDE Configuration Message Window during model configuration and should be deleted occasionally.
  - **`kernelMessageOutput*`** :
    Contain informational messages by the simulator kernel and CMAS, may be deleted unless needed for troubleshooting
  - **`<PID>_<date>_<time>.LOG`**
    `<PID>_<date>_<time>.ARCH_* :`
    Binary 'raw' log and archive files from a (most likely crashed) simulation session. Usually, they are converted to their _TXT counterparts (see above) after normal simulation shutdown by the CSS logger backend program and deleted.
  - **`ctg_server.message_log*, db_server.message_log*`** :
    Contain informational messages by the ctg– resp. db–server. Creation is usually disabled, but may be activeted for troubleshooting. Files may be deleted unless still needed for troubleshooting.
- **`$CSS_SIM_DIR`** (usually **`$HOME/css/simulations`**):
  This directory contains files named **`<hostname>@<modelname>@<PID>`**, describing currently active simulations, which are deleted during normal shutdown of a simulation. Check occasionally for files left by crashed simulations.
- **`$HOME/css/models`**:
  This directory contains simulation models that have been exported from the MDB into the file system, distributed over assorted subdirectories. The location of each model is described in the file **`contents.`**Exported models that are not needed any more should be deleted using the CSS Database Browser.

- $CSS_MODELS (usually $HOME/models.lib) :
  This directory tree contains all files (source code, Ada libraries, executables and runtime configuration files) related to CSS models that are controlled by the MDB. Paths in this tree are built as :
  `<element>–<mission>–<version>/library_sid_<cdu–sid>–V<version>.<issue>.<revision>–<test version>–<instance>/model_sid_<item–sid>`

  Note that most information in this directory tree is mandatory for the execution of a CSS simulation model and cannot be re–created once a model's CDU has been frozen in the MDB, so any model's subdirectory in this file system tree should only be deleted when definitely not needed any more (e.g. belonging to an obsolete system tree version).
- Additional model library directories :
  When using the Standalone–File–System version of CSS, arbitrary model library directories may be defined by the user, which are listed in the file **`$HOME/.cssdbrc`**. Models that are not needed any more should be deleted using the CSS File System Browser.

7.4.10 Troubleshooting

7.4.10.1 Model Editor : ’ORACLE Error’ or ’Database Crashed’ reported

This is most likely caused by problems in the MDB configuration. For more information, check:

- $MDA_HOME/data directory for a recent LOG*.css_db_server file
- if an ORACLE error code is reported by MDE or the MDA_HOME LOG file, the **`oerr`** utility may provide additional information : in a shell tool, enter :
  `oerr ora <error code>`
- If the problem is reprocducible, activation of CSS DB Server debugging output may provide additional information (see section 7.4.8). Add the following line to your ~/.user/cshrc script :
  `setenv CSS_DBS_OPTIONS –v4`
  This setting will log received requests from MDE and error messages to a file
  `$HOME/css/logger/db_server.message_log.<hostname>.<process–ID>`
  Reducing verbosity level to ’–v3’ will only log error messages, which may be a useful default setting, but will create lots of small log files in $HOME/css/logger. Higher levels should only be set on advice, since they will produce a huge amount of data.

When editing large models in the MDB (some hundred or more FB’s), contact you system administrator to make sure that sufficient ORACLE table space is available in the MDB (usually TS_MPS), temporary (usually TEMP) and large rollback (usually RS_LARGE) tablespaces. **As a rule of thumb, at least 20kBytes per function block should be available in each of these tablespaces.**

7.4.10.2 Problems during model configuration

- Check most recent `$HOME/css/logger/<model>_<user>_<architecture>_<date>_<time>.COMPILATION_REPORT_TXT` for error messages
- If the problem is reprocducible, activation of CSS CTG Server debugging output may provide additional information (see section 7.4.8). Add the following line to your ~/.user/cshrc script :
  `setenv CSS_LOGGING TRUE`
  This setting will log received requests from MDE, some execution tracing and error messages to a file
  `$HOME/css/logger/ctg_server.message_log.<user>.<hostname>.<process–ID>`.

7.4.10.3 Problems during model execution

- Check most recent `$HOME/css/logger/kernelMessageOutput<process_ID>` for error messages
- Check CMAS documentation for project specific adaptation system troubleshooting.

7.4.11 Creating Onboard End Items in the MDB

Note :
This section describes the required settings for configuration of the CGS standard CMAS.
Except for **Software Access Class** *and* **Software Type***, which are required for the creation of an onboard data reference in the model editor, these restrictions may not apply to project specific user defined MDB end items, which are to be handled by project specific CMAS configuration software (cf. section 7.4.7.1.40.3)*

7.4.11.1 End Item Type

The connection of a CSS simulator kernel to an external system (i.e. for a H/W in the loop simulation) is established by creating a number of properly defined onboard end items (stimuli and measurements) in the MDB that then can be referenced from the simulation model's interface items. This information is used by CSS to generate the simulation model specific configuration file allowing the Command and Measurement Adaptation System (CMAS) to initialize itself accordingly.

CGS provides a number of predefined end item types with stimulus resp. measurement characteristic, they can be created via the I_MDB tool. Note that end item types may be restricted to CDUs of specific domains, but by convention almost all end item types may be created in a CDU of domain CGS (e.g. EGSE specific end items may exist only in a CDU of domain EGSE or CGS). Onboard end items may be created directly as subnodes of a CDU; however, if there are many of them, it is usually better to organize them in a tree structure of virtual nodes.

If the end item types provided by CGS are not sufficient, new user defined end item types may be specified with the DADIMA tool prior to creation and installation of the MDB instance. To create a user defined end item type with stimulus or measurement characteristic, set its **Software Access Class** attribute to SEND for a stimulus type resp. READ for a measurement type; CSS will recognize the end item characteristics accordingly.

7.4.11.2 Mapping to CSS Data Type

A model input may reference a stimulus while a model output may reference a measurement. In both cases type compatibility between the referencing interface item and the referenced onboard end item has to be assured, i.e. CSS must be able to map the onboard end item to one of the CSS data types provided at the model interface. This is done by evaluating the item's engineering value type, and, for onboard end items of an integer engineering value type also the item's engineering range.

The engineering value type of a measurement is set when the end item type is specified with the DADIMA tool, it is given by the item's **Software Type** attribute. In contrast, the **Software Type** attribute of a stimulus is always NONE. The engineering value type of a stimulus is set when its formal parameter list (one formal parameter) is compiled with the CLS editor, i.e. it is derived from the UCL type of the formal parameter. An exception are stimuli of the predefined end item types PULSE_STIMULUS and BURST_PULSE_STIMULUS: They must not have a formal parameter (consequently the CLS editor cannot be invoked on them), their engineering value type is derived directly from the end item type.

The following table lists all possible engineering value types of onboard end items intended to be referenced by a CSS simulation model together with the corresponding CSS data types provided for simulation model interface items.

Onboard End Item Engineering Value Type	CSS Data Type
INTEGER_TYPE	SIGNED_BYTE SIGNED_SHORT_WORD SIGNED_INTEGER (depending on engineering range)

UNSIGNED_INTEGER_TYPE	UNSIGNED_BYTE UNSIGNED_SHORT_WORD UNSIGNED_INTEGER (depending on engineering range)
REAL_TYPE	REAL
LONG_REAL_TYPE	LONG_REAL
BOOLEAN_TYPE	BOOLEAN
STATE_CODE_TYPE	STATE_CODE
PULSE_TYPE	PULSE
BURST_PULSE_TYPE	BURST_PULSE

The following table lists all possible engineering value types of onboard end items intended to be referenced by a CSS simulation model together with the corresponding UCL types to be specified for the formal parameter of a stimulus end item.

Onboard End Item Engineering Value Type	UCL Type
INTEGER_TYPE	INTEGER
UNSIGNED_INTEGER_TYPE	UNSIGNED_INTEGER
REAL_TYPE	REAL
LONG_REAL_TYPE	LONG_REAL
BOOLEAN_TYPE	BOOLEAN
STATE_CODE_TYPE	STATECODE

For the predefined stimulus end item types, the CLS editor provides a default formal parameter list which should be compiled without any modification. However, for a stimulus of a user defined end item type, the formal parameter list must be specified explicitely as shown in the following example for a stimulus of engineering value type REAL_TYPE (if another engineering value type is desired, replace REAL by the appropriate UCL type):

```
(VALUE: REAL);
```

The pulse stimuli are treated specially, their engineering value type is derived directly from the end item type.

Onboard End Item Engineering Value Type	Onboard End Item Type
PULSE_TYPE	PULSE_STIMULUS
BURST_PULSE_TYPE	BURST_PULSE_STIMULUS

A mandatory aggregate for onboard end items of numeric engineering value type is an engineering range, i.e. the **Integer Engineering Range** for onboard end items of engineering value type INTEGER_TYPE, the **Unsigned Integer Engineering Range** for onboard end items of engineering value type UNSIGNED_INTEGER_TYPE resp. BURST_PULSE_TYPE, the **Float Engineering Range** for onboard end items of engineering value type REAL_TYPE resp. the **Double Float Engineering Range** for onboard end items of engineering value type LONG_REAL_TYPE.

In order to map an onboard end item of an integer engineering value type (i.e. engineering value type INTEGER_TYPE resp. UNSIGNED_INTEGER_TYPE) to a CSS data type, CSS evaluates the engineering range defined (i.e. the **Integer Engineering Range** resp. **Unsigned Integer Engineering Range**); the onboard end item is mapped to the smallest possible CSS data type. To avoid confusion, the engineering ranges should be defined as listed in the following table.

Engineering Range **Low Value**..**High Value**	CSS Data Type
–128..+127	SIGNED_BYTE
–32768..+32767	SIGNED_SHORT_WORD
–2147483648..+2147483647	SIGNED_INTEGER
0..255	UNSIGNED_BYTE
0..65535	UNSIGNED_SHORT_WORD
0..2147483647	UNSIGNED_INTEGER

The following table lists all CSS data types available for model interface items together with CGS predefined end item types that may be mapped to them. Note that for onboard end items of integer engineering value type (i.e. UNSIGNED_INTEGER_STIMULUS, UNSIGNED_INTEGER_MEASUREMENT, INTEGER_STIMULUS, EGSE_INTEGER_MEASUREMENT) the end item's engineering range has to be evaluated as described above to allow a non–ambiguous mapping.

CSS Data Type	Predefined Onboard End item Type
UNSIGNED_BYTE	UNSIGNED_INTEGER_STIMULUS UNSIGNED_INTEGER_MEASUREMENT
SIGNED_BYTE	INTEGER_STIMULUS EGSE_INTEGER_MEASUREMENT
UNSIGNED_SHORT_WORD	UNSIGNED_INTEGER_STIMULUS UNSIGNED_INTEGER_MEASUREMENT
SIGNED_SHORT_WORD	INTEGER_STIMULUS EGSE_INTEGER_MEASUREMENT
UNSIGNED_INTEGER	UNSIGNED_INTEGER_STIMULUS UNSIGNED_INTEGER_MEASUREMENT
SIGNED_INTEGER	INTEGER_STIMULUS EGSE_INTEGER_MEASUREMENT
REAL	EGSE_ANALOG_STIMULUS EGSE_FLOAT_MEASUREMENT
LONG_REAL	DOUBLE_FLOAT_STIMULUS DOUBLE_FLOAT_MEASUREMENT
BOOLEAN	BOOLEAN_STIMULUS BOOLEAN_MEASUREMENT
STATE_CODE	EGSE_DISCRETE_STIMULUS EGSE_DISCRETE_MEASUREMENT
PULSE	PULSE_STIMULUS
BURST_PULSE	BURST_PULSE_STIMULUS

The type definition of an onboard end item of engineering value type STATE_CODE must be completed by the specification of the statecode list. The mandatory list of the aggregate **Discrete Calibration** is used for this purpose (this aggregate comprises two attributes, the attribute **Discrete Calibration State Code** is used to specify a state code). Because the list of state codes is part of the type definition, it is relevant for type compatibility with the referencing simulation model's interface item. Both items must refer to the same state codes; however, the order in the respective lists doesn't matter.

The optional attribute **Engineering Units** of onboard end items with engineering value type REAL_TYPE resp. LONG_REAL_TYPE allows to refine the type definition by specifying an engineering unit. An engineering unit is relevant for type compatibility with the referencing simulation model's interface item; both items must refer to the same engineering unit.

7.4.11.3 Physical Address

The following attributes are mandatory (i.e. proper values have to be defined) for simple stimuli and measurements and for CCSDS TM/TC packets. For stimuli describing TC parameters (i.e. parameters contained in a TC packet) and measurements describing TM parameters (i.e. parameters contained in a TM packet), only the attributes **Remote Terminal Slot Class** and **Device Type** are applicable.

– **Remote Terminal Slot Class**
  ::= {BITS_32 | BITS_16 | BITS_8 | BITS_1}
    for a simple stimulus resp. measurement
  ::= PACKET
    for a CCSDS TM/TC packet or a stimulus resp. measurement describing a parameter of a CCSDS TM/TC packet
  ::= NON_STANDARD
    for an item transmitted via a Frontend Bus handled by a user defined non–standard driver

Note:
The Remote Terminal Slot Class must be the same for a specific Device Subaddress (Frontend Bus/Device Address/Device Subaddress), i.e. no different slot classes are allowed on one subaddress; this has to be assured for all onboard end items referenced by a specific simulation model.

Note:
On a specific Device Subaddress (Frontend Bus/Device Address/Device Subaddress), there may exist either only stimuli or only measurements; this has to be assured for all onboard end items referenced by a specific simulation model.

Note:
A complete Frontend Bus may be handled by a user defined non–standard driver linked with standard CMAS (e.g. it is not possible to declare a specific Device Subaddress as NON_STANDARD); the Frontend Bus must be configured to support NON_STANDARD (i.e. only NON_STANDARD) in the CMAS VME config file.

Note:
The onboard end item related information as specified in the MDB (i.e. physical address, packet definition, (de)calibration definition) is available also for a non–standard driver; however, it may or may not make use of this information, as appropriate.

– **Device Type**
  ::= MIL_BUS

– **Frontend Bus**
  ::= 0..2147483647
  logical bus (index) referred to in the CMAS VME config file

Note:
For each simulation model, the Frontend Busses handled must be indexed strictly sequential starting with 0 (i.e. 0, 1, 2, 3, ..., n–1 with n giving the number of Frontend Busses handled)

– **Device Address**
  ::= {0..30}
  MIL1553–Bus RT address (address 31 is reserved for broadcasting)

– **Device Subaddress**
  ::= {1..30}
  MIL1553–Bus RT subaddress (addresses 0 and 31 are reserved for mode codes allowing to command the RT itself)

– **Device Channel**
  ::= {0..15} for **Remote Terminal Slot Class** BITS_32
  ::= {0..31} for **Remote Terminal Slot Class** BITS_16
  ::= {0..63} for **Remote Terminal Slot Class** BITS_8
  ::= {0..511} for **Remote Terminal Slot Class** BITS_1 or NON_STANDARD
  ::= {0..0} for **Remote Terminal Slot Class** PACKET
  512 bits (i.e. 32 16–bit data words) is the maximum size of a single MIL–Bus message

The following attributes are optional for onboard end items of **Remote Terminal Slot Class** NON_STANDARD; they may be used to provide information for the control of a user defined non–standard driver (standard CMAS does not evaluate them).

– **Command 1**
  ::= STRING
  some arbitrary string

– **Command 2**
  ::= STRING
  some arbitrary string

7.4.11.4 CCSDS TM/TC Packets

A mandatory attribute of a stimulus describing a TC parameter or a measurement describing a TM parameter is the **Raw Value Size in Bits** (this attribute is not applicable for simple stimuli and measurements since the information is implicitely defined by their **Remote Terminal Slot Class** attribute). Such stimuli and measurements must be referenced from the packets they are contained in.

A TM packet (from CSS point of view a packet that is received by the external system from the simulator kernel) is described by an end item of type CCSDS_ADU_DESCRIPTION. Mandatory aggregates of this end item are the **CCSDS Primary Header** and the **Physical Address** (see above), an optional attribute is the **CCSDS Second Header**. The list of the aggregate **Measurement End Items** must contain an entry for each TM parameter. This aggregate comprises the attributes **End Item Reference**, i.e. a reference to a measurement end item, and **Location**, i.e. the bit offset in the CCSDS User Data Field (1..2147483647).

A TC packet (from CSS point of view a packet that is sent from the external system to the simulator kernel) is described by an end item of type EGSE_PREDEFINED_TC. Mandatory aggregates of this end item are the **CCSDS Primary Header** and the **Physical Address** (see above), an optional attribute is the **CCSDS Second Header**. The list of the aggregate **TC End Item References** must contain an entry for each TC parameter. This aggregate comprises the attributes **Stimulus Reference**, i.e. a reference to a stimulus end item, and **Location**, i.e. the bit offset in the CCSDS User Data Field (1..32768).

Note: Inputs of CSS simulation models can only be connected to (i.e. reference) stimulus end items. Originally the TC parameters of an EGSE_PREDEFINED_TC are specified by the CLS Editor together with a corresponding list of the aggregate **List of Parameters** (the command's parameters) and a list of the aggregate

General Bitstream Layout together with the corresponding lists of definition aggregates, i.e. **Integer Definition**, **Float Definition** and **Binary Definition** (the command's constant patterns). CSS will see neither the parameters nor the constant patterns. The **TC End Item References** list was introduced especially (and exclusively) for CSS. It allows to emulate the EGSE_PREDEFINED_TC's parameters and constant patterns redundantly with a number of separate stimulus end items. This list must comply exactly in engineering value types and locations (i.e. bit offsets) with the EGSE_PREDEFINED_TC's parameters and constant patterns as specified by the aggregates listed above.

7.4.11.5 Calibration/Decalibaration Definition

The meaning of calibration and decalibration is as follows:

Calibration
transformation of a raw value into an engineering value

Decalibration
transformation of an engineering value into a raw value

From the Ground System's point of view, stimuli have to be decalibrated while measurements have to be calibrated (CMAS performs the inverse operation for CSS: decalibration of measurements, calibration of stimuli).

<u>Numeric onboard end items</u> (i.e. end items of engineering value type INTEGER_TYPE, UNSIGNED_INTEGER_TYPE, REAL_TYPE, LONG_REAL_TYPE or BURST_PULSE_TYPE)

The following attributes are mandatory (i.e. proper values have to be defined):

– **Raw Value Type**
  ::= SIGNED_INTEGER for onboard end items of engineering value type INTEGER_TYPE
  ::= UNSIGNED_INTEGER for onboard end items of engineering value type
  UNSIGNED_INTEGER_TYPE or BURST_PULSE_TYPE
  ::= {UNSIGNED_INTEGER | SIGNED_INTEGER} for onboard end items of engineering value
  type REAL_TYPE or LONG_REAL_TYPE

– **Calib Curve Type**
  ::= {POINT_PAIRS | POLYNOM | IDENTICAL}

A mandatory aggregate is the raw value range, i.e. the **Integer Raw Value Range** for an item with **Raw Value Type** SIGNED_INTEGER resp. the **Unsigned Integer Raw Value Range** for an item with **Raw Value Type** UNSIGNED_INTEGER.

Another mandatory aggregate is the (de)calibration definition for end items with **Calib Curve Type** POINT_PAIRS or POLYNOM, i.e. **Analog Decalibration Point Pairs** for a stimulus end item with **Calib Curve Type** POINT_PAIRS, **Analog Point Pairs** for a measurement end item with **Calib Curve Type** POINT_PAIRS, **Analog Decalibration Coefficients** for a stimulus end item with **Calib Curve Type** POLYNOM resp. **Analog Calibration Coefficients** for a measurement end item with **Calib Curve Type** POLYNOM.

Note:
The (de)calibaration definition must allow inversion, i.e. each raw value must be associated to exactly one engineering value and vice versa (unique relation raw value–engineering value).

Note:
The (de)calibaration definition must assure that the range limits computed (raw value range limits

resp. engineering range limits) match exactly the respective range limit specification, which may be tricky for polynom (de)calibration because of possible rounding errors.

An optional aggregate for end items with **Calib Curve Type** POLYNOM is the (de)calibration point pairs allowing to determine the conversion in inverse direction, i.e. **Analog Point Pairs** for a stimulus end item resp. **Analog Decalibration Point Pairs** for a measurement end item. If not explicitely specified, point pairs for the conversion in inverse direction will be calculated automatically from the polynom.

<u>Discrete onboard end items</u> (i.e. end items of engineering value type STATE_CODE_TYPE)

The following attribute and aggregate are mandatory:

– **Raw Value Type**
  ::= {UNSIGNED_INTEGER | INTEGER}

– **Discrete Calibration**
  a list of the aggregate **Discrete Calibration**; this aggregate comprises the attributes **Discrete Calibration State Code** and **Discrete Calibration Raw Value**, i.e. a state code specification and the associated raw value

Another mandatory aggregate is the raw value range, i.e. the **Integer Raw Value Range** for an item with **Raw Value Type** SIGNED_INTEGER resp. the **Unsigned Integer Raw Value Range** for an item with **Raw Value Type** UNSIGNED_INTEGER.

<u>Boolean onboard end items</u> (i.e. end items of engineering value type BOOLEAN_TYPE or PULSE_TYPE)

The following attribute and aggregate are mandatory:

– **Raw Value Type**
  ::= {UNSIGNED_INTEGER | INTEGER}

– **Boolean Calibration**
  this aggregate comprises the attributes **True** and **False** allowing to specify the respective raw values

7.4.11.6 Feedback

When an onboard end item is selected in the CSS Database Browser, CSS provides its characteristic (stimulus resp. measurement), the CSS data type it is mapped to (NONE if mapping is not possible), and, if the definition of an onboard end item is erroneous (e.g. there are mandatory attributes missing), also an error message (e.g. no raw value type, no raw value range, no calibration description, no coefficients, no physical address, etc.).

7.4.12 Description of mathematical constants and routines delivered with CSS

—
— KEYWORDS — CSS_MATH
— Project CSS
— ADD, section 5.3 (CSS Repository)
— HOOD Object CSS_MATH
—

This description shall support the user’s model SW development. The ”undermentioned” mathematical constants and routines may be used within the AIL (Atomic Implementation Language, i.e. Ada subset) of a synchronous or asynchronous Function Block and are annotated as far as necessary.

— CONSTANTS

Values have been taken from the following sources:

- CRC Handbook of Tables for Mathematics, Fourth Ed., Robert C. Weast (ed.), 1970, The Chemical Rubber Co.
- Knuth, Donald E., ’Fundamental Algorithms’, Vol. 1 of ’The Art of Computer Programming’, 2nd ed., 1973. (Appendix B).
- Davis, Harold T. and Fisher, Vera J., ’Arithmetical Tables’, Vol. III of ’Tables of the Mathematical Functions’, The Principia Press, Texas, 1962.
- Fletcher, A. et al., ’An Index of Mathematical Tables’, Scientific Computing Service Limited, London, 1962.

Where values exist in more than one source, such values have been cross checked. In all cases, such values agree except for possibly a value of one in the last digit. In such cases of difference, the higher value is used, under the assumption that it is a rounded value and that the lower value is a truncated value.

The list of constants specified is a combination of lists in the various sources and the list in the Elementary Function Package Proposal. Multiples and inverses of constants, which are computable at compile time with full accuracy, have been eliminated. The only exceptions are 1.0/pi and 1.0/e which are so frequently encountered that their definition is virtually ”forced.”

Naming is a combination of suggested names in the Elementary Function Package Proposal, non–conflict with likely variable names (single letter disallowed), descriptive naming (for in–use recognition), and a length decision. Names are shorter for constants that are more frequently used and more likely to be known. The naming criteria sometimes conflict with the desire for naming consistency.

Hence we have BASE_E (no single letter names) but E is used in all combination names. Similarly, GOLDEN_RATIO is spelled out (PHI likely to conflict with user variable names), but PHI is used for all composite names. The hardest decision was choosing BIN_LOG.., NAT_LOG.., and COM_LOG.. over the often used LN2, LN, and LOG.

```
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------

--   Basic constants :
--
--   pi, e, 1/pi, 1/e :
--
PI        : constant :=  3.14159_26535_89793_23846_26433_83279_50288;
BASE_E    : constant :=  2.71828_18284_59045_23536_02874_71352_66250;
INV_PI    : constant :=  0.31830_98861_83790_67153_77675_26745_02872;
INV_E     : constant :=  0.36787_94411_71442_32159_55237_70161_46087;
--
--   Square roots :
--
SQRT_2    : constant :=  1.41421_35623_73095_04880_16887_24209_69808;
SQRT_3    : constant :=  1.73205_08075_68877_29352_74463_41505_87237;
SQRT_5    : constant :=  2.23606_79774_99789_69640_91736_68731_27624;
SQRT_7    : constant :=  2.64575_13110_64590_59050_16157_53639_26043;
SQRT_10   : constant :=  3.16227_76601_68379_33199_88935_44432_71853;
SQRT_PI   : constant :=  1.77245_38509_05516_02729_81674_83341_14518;
--
--   Cube and fourth roots :
--
CBRT_2    : constant :=  1.25992_10498_94873_16476_72106_07278_22835;
CBRT_3    : constant :=  1.44224_95703_07408_38232_16383_10780_10959;
CBRT_4    : constant :=  1.58740_10519_68199_47475;
CBRT_10   : constant :=  2.15443_46900_31883_72715;
FOURTH_RT_2    : constant :=  1.18920_71150_02721_06671_74999_70560_47592;
FOURTH_RT_10   : constant :=  1.77827_94100_38922_80122;
--
--
--   Common expressions with pi and e :
--
--  pi**2, pi**e, e**2, e**e, e**(1/e), e**(-e), e**(-1/e),
--  e**pi, e**(pi/2), e**(pi/4); e**(-pi), e**(-pi/2), e**(-p/4)
--  Note: i**i = e**(-pi/2) where i is the unit imaginary value
--
PI_SQ     : constant :=  9.86960_44010_89358_61883_44909_99876_15114;
PI_TO_E   : constant := 22.45915_77183_61045_47342_71522_045;
EXP_SQ    : constant :=  7.38905_60989_30650_22723_04274_60575_00781;
EXP_E     : constant := 15.15426_22414_79264;
EXP_INV_E : constant :=  1.44466_78610_09766;
EXP_NEG_E : constant :=  0.06598_80358_45312;
EXP_NEG_INV_E : constant :=  0.69220_06275_55346;
EXP_PI    : constant := 23.14069_26327_79269_00572_90864;
EXP_HALF_PI    : constant :=  4.81047_73809_65351_65547_30357;
EXP_4TH_PI     : constant :=  2.19328_00507_38015_45655_97696_59278_73822;
EXP_NEG_PI     : constant :=  0.04321_39182_63772_24977_44177;
EXP_NEG_HALF_PI     : constant :=  0.20787_95763_50761_90854_69556;
EXP_NEG_4TH_PI : constant :=  0.45593_81277_65996_23676_59212;
--
```

```
--   Base e logarithms :
--
NAT_LOG_2     : constant :=  0.69314_71805_59945_30941_72321_21458_17657;
NAT_LOG_3     : constant :=  1.09861_22886_68109_69139_52452_36922_52570;
NAT_LOG_10    : constant :=  2.30258_50929_94045_68401_79914_54684_36421;
NAT_LOG_E     : constant :=  1.00000_00000_00000_00000_00000_00000_00000;
NAT_LOG_PI    : constant :=  1.14472_98858_49400_17414_34273_51353_05871;
--
--   Base 10 logarithms :
--
COM_LOG_2     : constant :=  0.30102_99956_63981_19521_37388_94724_49303;
COM_LOG_3     : constant :=  0.47712_12547_19662_43729_50279_03255_11531;
COM_LOG_10    : constant :=  1.00000_00000_00000_00000_00000_00000_00000;
COM_LOG_E     : constant :=  0.43429_44819_03251_82765_11289_18916_60508;
COM_LOG_PI    : constant :=  0.49714_98726_94133_85435_12682_88290_89887;
--
--   Base 2 logarithms :
--
BIN_LOG_2     : constant :=  1.00000_00000_00000_00000_00000_00000_00000;
BIN_LOG_3     : constant :=  NAT_LOG_3  / NAT_LOG_2 ;
BIN_LOG_10    : constant :=  NAT_LOG_10 / NAT_LOG_2 ;
BIN_LOG_E     : constant :=  1.0        / NAT_LOG_2 ;
BIN_LOG_PI    : constant :=  NAT_LOG_PI / NAT_LOG_2 ;
--
--   Golden ratio, phi ( a/b where a/b = b/(a-b) ) :
--
GOLDEN_RATIO  : constant :=  1.61803_39887_49894_84820_45868_34365_63812;
NAT_LOG_PHI   : constant :=  0.48121_18250_59603_44749_77589_13424_36842;
--
--   Euler's constant (gamma), log(base e) gamma, e**gamma, e**(-gamma) :
--
GAMMA         : constant :=  0.57721_56649_01532_86060_65120_90082_40243;
NAT_LOG_GAMMA : constant := -0.54953_93129_81644_82233_7662 ;
COM_LOG_GAMMA : constant := -1.76133_81087_83167_61054_0;
EXP_GAMMA     : constant :=  1.78107_24179_90197_98523_65041_03107_17955;
EXP_NEG_GAMMA : constant :=  0.56145_94835_66885_16983;
--
--   Trigonometric and hyperbolic values for 1.0 radian :
--
SIN_1         : constant :=  0.84147_09848_07896_50665_25023_21630_29900;
COS_1         : constant :=  0.54030_23058_68139_71740_09366_07442_97660;
TAN_1         : constant :=  1.55740_77246_54902;
SINH_1        : constant :=  1.17520_11936_43801;
COSH_1        : constant :=  1.54308_06348_15244;
TANH_1        : constant :=  0.76159_41559_55764;
--
```

```
-----------------------------------------------------------------
--  EXCEPTIONS
-----------------------------------------------------------------
--

DOMAIN_ERROR  : raised if UNIX errno returns EDOM, indicating the arguments were
not valid for the function
RANGE_ERROR   : raised if UNIX errno returns ERANGE, indicating that the correct
value cannot be computed

--
-----------------------------------------------------------------
--  FUNCTION SPECIFICATIONS
-----------------------------------------------------------------
--

-- Exponential, logarithm and root routines :
--
-- Long_Float versions :
--
function EXP (X : LONG_REAL) return LONG_REAL;      -- e**x
    -- raises RANGE_ERROR if correct value would overflow

function NAT_LOG (X : LONG_REAL) return LONG_REAL;  -- log base e (x)
function LOG(X : LONG_REAL) return LONG_REAL renames NAT_LOG;
    -- raises DOMAIN_ERROR if x is zero or negative

function COM_LOG (X : LONG_REAL) return LONG_REAL;  -- log base 10 (x)

function BIN_LOG (X : LONG_REAL) return LONG_REAL;  -- log base 2 (x)
    -- raises DOMAIN_ERROR if x is zero or negative

function "**" (X,Y : LONG_REAL) return LONG_REAL;   -- x**y
    -- raises RANGE_ERROR if correct value would overflow
    -- raises DOMAIN_ERROR when first argument is negative and second is
    -- non-integer or when first argument is 0 and second is <= 0

function SQRT (X : LONG_REAL) return LONG_REAL;     -- x**(1/2)
    -- raises DOMAIN_ERROR when x is negative

--
--  Short_float versions :
--

function EXP (X : REAL) return REAL;
    -- raises RANGE_ERROR if correct value would overflow

function NAT_LOG (X : REAL) return REAL;
function LOG(X : REAL) return REAL renames NAT_LOG;
    -- raises DOMAIN_ERROR if x is zero or negative
```

```
function COM_LOG (X : REAL) return REAL;

function BIN_LOG (X : REAL) return REAL;
    –– raises DOMAIN_ERROR if x is zero or negative

function ”**” (X,Y : REAL) return REAL;
    –– raises RANGE_ERROR if correct value would overflow
    –– raises DOMAIN_ERROR when first argument is negative and second is
    –– non–integer or when first argument is 0 and second is <= 0

function SQRT (X : REAL) return REAL;
    –– raises DOMAIN_ERROR when x is negative


––
––  Floor and ceiling functions :

––
–– Long_Float versions :
––
function FLOOR (X : LONG_REAL) return LONG_REAL;
function CEIL (X : LONG_REAL) return LONG_REAL;

––
–– Short_float versions :

––
function FLOOR (X : REAL) return REAL;
function CEIL (X : REAL) return REAL;



––
––  Gamma functions :

––
type SIGN_TYPE is (NEGATIVE, POSITIVE);
––
–– Long_Float versions :
––
type LONG_REAL_GAMMA_COMPONENTS is record
    LOG_OF_GAMMA : LONG_REAL;
    SIGN : SIGN_TYPE;
    end record;


function CALC_GAMMA(X : LONG_REAL) return LONG_REAL;
    –– raises DOMAIN_ERROR when x <= 0 and x is an integer

function LOG_GAMMA(X : LONG_REAL) return LONG_REAL_GAMMA_COMPONENTS;
    –– raises DOMAIN_ERROR when x <= 0 and x is an integer

––
–– Short_float versions :
```

```
--
type REAL_GAMMA_COMPONENTS is record
    LOG_OF_GAMMA : REAL;
    SIGN : SIGN_TYPE;
    end record;

function CALC_GAMMA (X : REAL) return REAL;
    -- raises DOMAIN_ERROR when x <= 0 and x is an integer
function LOG_GAMMA(X : REAL) return REAL_GAMMA_COMPONENTS;
    -- raises DOMAIN_ERROR when x <= 0 and x is an integer
```

--
-- Rectangular to polar coordinates functions :
--

Note:
ANGLE *returns radians result, range –pi/2 .. +pi/2.* ***ANGLE*** *is ARCTAN (1st_arg/2nd_arg). The call* ***ANGLE (yc,xc)*** *(where yc is the y–axis component and xc is the x–axis component) yields the angle from the x–axis to the complex value specified by (xc,yc). The call* ***ANGLE (x=>xc,y=>yc)*** *is identical to the above call.*

```
--
-- Long_Float versions :
--
function RADIUS (X,Y : LONG_REAL) return LONG_REAL;

function ANGLE (Y,X : LONG_REAL) return LONG_REAL;

--
-- Short_float versions :
--

function RADIUS (X,Y : REAL) return REAL;
function ANGLE (Y,X : REAL) return REAL;
```

--
-- Bessel functions :--
--

```
-- Long_Float versions :
--
function J0 (X : LONG_REAL) return LONG_REAL;
function J1 (X : LONG_REAL) return LONG_REAL;
function JN (N : INTEGER; X : LONG_REAL) return LONG_REAL;

function Y0 (X : LONG_REAL) return LONG_REAL;
    -- raises DOMAIN_ERROR when x is negative

function Y1 (X : LONG_REAL) return LONG_REAL;
    -- raises DOMAIN_ERROR when x is negative

function YN (N : INTEGER; X : LONG_REAL) return LONG_REAL;
    -- raises DOMAIN_ERROR when x is negative
```

```
--
-- Short_float versions :
--

function J0 (X : REAL) return REAL;
function J1 (X : REAL) return REAL;
function JN (N : INTEGER; X : REAL) return REAL;

function Y0 (X : REAL) return REAL;
    -- raises DOMAIN_ERROR when x is negative

function Y1 (X : REAL) return REAL;
    -- raises DOMAIN_ERROR when x is negative

function YN (N : INTEGER; X : REAL) return REAL;
    -- raises DOMAIN_ERROR when x is negative


--
--     Basic trigonometric functions :
--
--  NOTE:  Inputs are radians, outputs are unit-less ratios

--
-- Long_Float versions :
--

function SIN (X : LONG_REAL) return LONG_REAL;
function COS (X : LONG_REAL) return LONG_REAL;

function TAN (X : LONG_REAL) return LONG_REAL;
    -- raises RANGE_ERROR for singular points
    -- value of TAN is garbage for x > 2**31

--
-- Short_float versions :
--

function SIN (X : REAL) return REAL;

function COS (X : REAL) return REAL;

function TAN (X : REAL) return REAL;
    -- raises RANGE_ERROR for singular points
    -- value of TAN is garbage for x > 2**31


--
--     Basic inverse trigonometric functions :
--
--  NOTE: Inputs are unit-less ratios, outputs are radians

--
-- Long_Float versions :
--
```

```
function ARCSIN (X : LONG_REAL) return LONG_REAL;
    -- -1 <= x ,= +1
    -- range is -pi/2 .. +pi/2
    -- raises DOMAIN_ERROR when x > 1
function ARCCOS (X : LONG_REAL) return LONG_REAL;
    -- -1 <= x ,= +1
    -- range is 0 .. +pi
    -- raises DOMAIN_ERROR when x > 1

function ARCTAN (X : LONG_REAL) return LONG_REAL;
function ATAN (X : LONG_REAL) return LONG_REAL renames ARCTAN;
    -- x is unbounded
    -- range is -pi/2 .. +pi/2

--
-- Short_float versions :
--

function ARCSIN (X : REAL) return REAL;
    -- -1 <= x ,= +1
    -- range is -pi/2 .. +pi/2
    -- raises DOMAIN_ERROR when x > 1

function ARCCOS (X : REAL) return REAL;
    -- -1 <= x ,= +1
    -- range is 0 .. +pi
    -- raises DOMAIN_ERROR when x > 1

function ARCTAN (X : REAL) return REAL;
function ATAN (X : REAL) return REAL renames ARCTAN;
    -- x is unbounded
    -- range is -pi/2 .. +pi/2


--
--    Basic inverse trigonometric functions :
--
-- Long_Float versions :
--

function SINH (X : LONG_REAL) return LONG_REAL;
    -- returns huge value with appropriate sign when correct value would overflow

function COSH (X : LONG_REAL) return LONG_REAL;
    -- returns huge value with appropriate sign when correct value would overflow

function TANH (X : LONG_REAL) return LONG_REAL;

--
-- Short_float versions :
--

function SINH (X : REAL) return REAL;
    -- returns huge value with appropriate sign when correct value would overflow
```

```
function COSH (X : REAL) return REAL;
    –– returns huge value with appropriate sign when correct value would overflow

function TANH (X : REAL) return REAL;


–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

––
––
–– DISTRIBUTION AND COPYRIGHT :
––
This software is copyright 1995 by DASA–RI, RIT14, Bremen, Germany. All rights re-
served.  No part of the material protected by this copyright notice may be repro-
duced or utilized in any form without written permission of the copyright owner.
––
––
```

7.4.13 CSS Data Types

```
-----------------------------------------------------------------------------
--
-- ABSTRACT -- CSS_TYPES
-- Provides the set of types defined for CSS MDE model variables.

-------------------------------------------------------------------------------
--
-- KEYWORDS -- CSS_TYPES
-- Project CSS-Downsizing
-- <document refs; e.g.: DDD section 3.2>
-- HOOD object CSS_TYPES
-- MDE model variable types

-------------------------------------------------------------------------------
--
-- CONTENTS -- CSS_TYPES
--
-- Type:      Package Spec (+ implicit body)
--
-- Version:
-- 2.1, 10.11.93, J.Hoyng: initial version
-- 3.1, 21.07.94, J.Hoyng: the TIME type depends now on SMT_CALENDAR.TIME
--                  deleted type declaration DURATION
-- 3.2, 06.04.95, J.Hoyng: implemented new types
-- 4.1, 23.08.95, J.Hoyng: build_2 version
--
-- Purpose:
--   Provide CSS defined typeset
--
-- External:
-- From Ada system:
--   CALENDAR
--
-- From CGS:
--   NUMERIC_TYPES
--
-- From CSS:
--   DB_DEFINITIONS
--      SMT_CALENDAR
--
-- Machine Dependencies:
--   none
--
-- Compiler Dependencies:
--   none
-------------------------------------------------------------------------------

£
-- imports from Ada system:
with CALENDAR;
```

```
–– imports from CGS:
with NUMERIC_TYPES;

–– imports from CSS:
with DB_DEFINITIONS;
with SMT_CALENDAR;

package CSS_TYPES is

type CSS_SW_TYPES is (
         NONE,
         UNSIGNED_BYTE_TYPE,
         SIGNED_BYTE_TYPE,
         UNSIGNED_SHORT_WORD_TYPE,
         SIGNED_SHORT_WORD_TYPE,
         UNSIGNED_INTEGER_TYPE,
         SIGNED_INTEGER_TYPE,
         REAL_TYPE,
         LONG_REAL_TYPE,
         COMPLEX_TYPE,
         BOOLEAN_TYPE,
         TIME_TYPE,
         LONG_DURATION_TYPE,
         STATE_CODE_TYPE,
         PULSE_TYPE,
         BURST_PULSE_TYPE,
         UNSIGNED_BYTE_VECTOR_TYPE,
         SIGNED_BYTE_VECTOR_TYPE,
         UNSIGNED_SHORT_WORD_VECTOR_TYPE,
         SIGNED_SHORT_WORD_VECTOR_TYPE,
         UNSIGNED_INTEGER_VECTOR_TYPE,
         SIGNED_INTEGER_VECTOR_TYPE,
         REAL_VECTOR_TYPE,
         LONG_REAL_VECTOR_TYPE,
         COMPLEX_VECTOR_TYPE,
         BOOLEAN_VECTOR_TYPE,
         UNSIGNED_BYTE_MATRIX_TYPE,
         SIGNED_BYTE_MATRIX_TYPE,
         UNSIGNED_SHORT_WORD_MATRIX_TYPE,
         SIGNED_SHORT_WORD_MATRIX_TYPE,
         UNSIGNED_INTEGER_MATRIX_TYPE,
         SIGNED_INTEGER_MATRIX_TYPE,
         REAL_MATRIX_TYPE,
         LONG_REAL_MATRIX_TYPE,
         COMPLEX_MATRIX_TYPE,
         BOOLEAN_MATRIX_TYPE,
         BYTE_STREAM_TYPE,
         LAST_TYPE);

–– all CSS_SW_TYPES used during runtime
subtype CSS_RUNTIME_SW_TYPES is CSS_SW_TYPES range
      CSS_SW_TYPES’SUCC(NONE) .. CSS_SW_TYPES’PRED (LAST_TYPE);
```

```
type UNSIGNED_INTEGER is new NUMERIC_TYPES.UNSIGNED_INTEGER32
                    range 0 .. NUMERIC_TYPES.UNSIGNED_INTEGER32'LAST;

------------------------------
-- SIGNED_BYTE
------------------------------
type SIGNED_BYTE is new NUMERIC_TYPES.INTEGER8;

SIGNED_BYTE_INITIAL_VALUE : constant SIGNED_BYTE := 0;

type SIGNED_BYTE_VECTOR is
       array (UNSIGNED_INTEGER range <>) of SIGNED_BYTE;

type SIGNED_BYTE_VECTOR_ACCESS is access SIGNED_BYTE_VECTOR;

type SIGNED_BYTE_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of SIGNED_BYTE;

type SIGNED_BYTE_MATRIX_ACCESS is access SIGNED_BYTE_MATRIX;

------------------------------
--  UNSIGNED_BYTE
------------------------------
type UNSIGNED_BYTE is new NUMERIC_TYPES.BYTE;

UNSIGNED_BYTE_INITIAL_VALUE : constant UNSIGNED_BYTE := 0;

type UNSIGNED_BYTE_VECTOR is array (UNSIGNED_INTEGER range <>) of UNSIGNED_BYTE;

type UNSIGNED_BYTE_VECTOR_ACCESS is access UNSIGNED_BYTE_VECTOR;

type UNSIGNED_BYTE_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of UNSIGNED_BYTE;

type UNSIGNED_BYTE_MATRIX_ACCESS is access UNSIGNED_BYTE_MATRIX;

------------------------------
--  UNSIGNED_SHORT_WORD
------------------------------
type UNSIGNED_SHORT_WORD is new NUMERIC_TYPES.UNSIGNED_INTEGER16;

UNSIGNED_SHORT_WORD_INITIAL_VALUE : constant UNSIGNED_SHORT_WORD := 0;

type UNSIGNED_SHORT_WORD_VECTOR is
       array (UNSIGNED_INTEGER range <>) of UNSIGNED_SHORT_WORD;

type UNSIGNED_SHORT_WORD_VECTOR_ACCESS is access UNSIGNED_SHORT_WORD_VECTOR;

type UNSIGNED_SHORT_WORD_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of UNSIGNED_SHORT_WORD;

type UNSIGNED_SHORT_WORD_MATRIX_ACCESS is access UNSIGNED_SHORT_WORD_MATRIX;

------------------------------
-- SIGNED_SHORT_WORD
------------------------------
type SIGNED_SHORT_WORD is new NUMERIC_TYPES.INTEGER16;
```

```
SIGNED_SHORT_WORD_INITIAL_VALUE : constant SIGNED_SHORT_WORD := 0;

type SIGNED_SHORT_WORD_VECTOR is
        array (UNSIGNED_INTEGER range <>) of SIGNED_SHORT_WORD;

type SIGNED_SHORT_WORD_VECTOR_ACCESS is access SIGNED_SHORT_WORD_VECTOR;

type SIGNED_SHORT_WORD_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of SIGNED_SHORT_WORD;

type SIGNED_SHORT_WORD_MATRIX_ACCESS is access SIGNED_SHORT_WORD_MATRIX;

------------------------------
-- SIGNED_INTEGER
------------------------------
type SIGNED_INTEGER is new NUMERIC_TYPES.INTEGER32;

SIGNED_INTEGER_INITIAL_VALUE : constant SIGNED_INTEGER := 0;

type SIGNED_INTEGER_VECTOR is
       array (UNSIGNED_INTEGER range <>) of SIGNED_INTEGER;

type SIGNED_INTEGER_VECTOR_ACCESS is access SIGNED_INTEGER_VECTOR;

type SIGNED_INTEGER_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of SIGNED_INTEGER;

type SIGNED_INTEGER_MATRIX_ACCESS is access SIGNED_INTEGER_MATRIX;

------------------------------
--  UNSIGNED_INTEGER
------------------------------

UNSIGNED_INTEGER_INITIAL_VALUE : constant UNSIGNED_INTEGER := 0;

type UNSIGNED_INTEGER_VECTOR is
       array (UNSIGNED_INTEGER range <>) of UNSIGNED_INTEGER;

type UNSIGNED_INTEGER_VECTOR_ACCESS is access UNSIGNED_INTEGER_VECTOR;

type UNSIGNED_INTEGER_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of UNSIGNED_INTEGER;

type UNSIGNED_INTEGER_MATRIX_ACCESS is access UNSIGNED_INTEGER_MATRIX;

------------------------------
-- REAL
------------------------------
type REAL is new NUMERIC_TYPES.SINGLE_FLOAT;

REAL_INITIAL_VALUE : constant REAL := 0.0;

type REAL_VECTOR is
       array (UNSIGNED_INTEGER range <>) of REAL;

type REAL_VECTOR_ACCESS is access REAL_VECTOR;
```

```
type REAL_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of REAL;

type REAL_MATRIX_ACCESS is access REAL_MATRIX;

------------------------------
-- LONG_REAL
------------------------------
type LONG_REAL is new NUMERIC_TYPES.DOUBLE_FLOAT;

LONG_REAL_INITIAL_VALUE : constant LONG_REAL := 0.0;

type LONG_REAL_VECTOR is
       array (UNSIGNED_INTEGER range <>) of LONG_REAL;

type LONG_REAL_VECTOR_ACCESS is access LONG_REAL_VECTOR;

type LONG_REAL_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of LONG_REAL;

type LONG_REAL_MATRIX_ACCESS is access LONG_REAL_MATRIX;

------------------------------
-- COMPLEX
------------------------------

type COMPLEX is
  record
    RE, IM :  LONG_REAL;
  end record;

COMPLEX_INITIAL_VALUE : constant COMPLEX := (0.0, 0.0);

type COMPLEX_VECTOR is
       array (UNSIGNED_INTEGER range <>) of COMPLEX;

type COMPLEX_VECTOR_ACCESS is access COMPLEX_VECTOR;

type COMPLEX_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of COMPLEX;

type COMPLEX_MATRIX_ACCESS is access COMPLEX_MATRIX;

------------------------------
-- BOOLEAN
------------------------------
BOOLEAN_INITIAL_VALUE : constant BOOLEAN := TRUE;

type BOOLEAN_VECTOR is
       array (UNSIGNED_INTEGER range <>) of BOOLEAN;

type BOOLEAN_VECTOR_ACCESS is access BOOLEAN_VECTOR;

type BOOLEAN_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of BOOLEAN;
```

```
type BOOLEAN_MATRIX_ACCESS is access BOOLEAN_MATRIX;

------------------------------
-- LONG_DURATION
------------------------------
subtype LONG_DURATION is SMT_CALENDAR.LONG_DURATION;

    function "+"(RIGHT: LONG_DURATION) return LONG_DURATION
               renames SMT_CALENDAR."+";
    function "-"(RIGHT: LONG_DURATION) return LONG_DURATION
               renames SMT_CALENDAR."-";
    function "abs"(RIGHT: LONG_DURATION) return LONG_DURATION
               renames SMT_CALENDAR."abs";
    function "+"(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return LONG_DURATION
               renames SMT_CALENDAR."+";
    function "-"(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return LONG_DURATION
               renames SMT_CALENDAR."-";
    function "*"(LEFT: LONG_DURATION; RIGHT: INTEGER) return LONG_DURATION
               renames SMT_CALENDAR."*";
--    function "*"(LEFT: INTEGER; RIGHT: LONG_DURATION) return LONG_DURATION
--               renames SMT_CALENDAR.MUL;
    function "/"(LEFT: LONG_DURATION; RIGHT: INTEGER) return LONG_DURATION
               renames SMT_CALENDAR."/";

    function "="(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return BOOLEAN
               renames SMT_CALENDAR."=";
    function "<"(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return BOOLEAN
               renames SMT_CALENDAR."<";
    function "<="(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return BOOLEAN
               renames SMT_CALENDAR."<=";
    function ">"(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return BOOLEAN
               renames SMT_CALENDAR.">";
    function ">="(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return BOOLEAN
               renames SMT_CALENDAR.">=";

LONG_DURATION_INITIAL_VALUE : constant LONG_DURATION := 0.0;

------------------------------
-- TIME
------------------------------
subtype TIME is SMT_CALENDAR.TIME;

    subtype YEAR_NUMBER    is INTEGER        range 1901 .. 2099;
    subtype MONTH_NUMBER   is INTEGER        range 1 .. 12;
    subtype DAY_NUMBER     is INTEGER        range 1 .. 31;
    subtype DAY_DURATION   is LONG_DURATION  range 0.0 .. 86_400.0;

    function CLOCK return TIME renames SMT_CALENDAR.CLOCK;
```

```
    function YEAR   (DATE: TIME) return YEAR_NUMBER renames SMT_CALENDAR.YEAR;
    function MONTH  (DATE: TIME) return MONTH_NUMBER renames SMT_CALENDAR.MONTH;
    function DAY    (DATE: TIME) return DAY_NUMBER renames SMT_CALENDAR.DAY;
    function SECONDS(DATE: TIME) return DAY_DURATION renames SMT_CALENDAR.SECONDS;
    procedure SPLIT (DATE   : in     TIME;
                     YEAR   : out    YEAR_NUMBER;
                     MONTH  : out    MONTH_NUMBER;
                     DAY    : out    DAY_NUMBER;
                     SECONDS: out    DAY_DURATION) renames SMT_CALENDAR.SPLIT;

    function TIME_OF(YEAR   : YEAR_NUMBER;
                     MONTH  : MONTH_NUMBER;
                     DAY    : DAY_NUMBER;
                     SECONDS: DAY_DURATION := 0.0) return TIME
                                  renames SMT_CALENDAR.TIME_OF;

    function "+"    (LEFT: TIME;        RIGHT: LONG_DURATION)    return TIME
                                 renames SMT_CALENDAR."+";
    function "+"    (LEFT: LONG_DURATION;    RIGHT: TIME)    return TIME
                                 renames SMT_CALENDAR."+";
    function "–"    (LEFT: TIME;        RIGHT: LONG_DURATION)    return TIME
                                 renames SMT_CALENDAR."–";
    function "–"    (LEFT: TIME;        RIGHT: TIME)    return LONG_DURATION
                                 renames SMT_CALENDAR."–";

    function "="    (LEFT, RIGHT: TIME) return BOOLEAN
                                 renames SMT_CALENDAR."=";
    function "<"    (LEFT, RIGHT: TIME) return BOOLEAN
                                 renames SMT_CALENDAR."<";
    function "<="   (LEFT, RIGHT: TIME) return BOOLEAN
                                 renames SMT_CALENDAR."<=";
    function ">"    (LEFT, RIGHT: TIME) return BOOLEAN
                                 renames SMT_CALENDAR.">";
    function ">="   (LEFT, RIGHT: TIME) return BOOLEAN
                                 renames SMT_CALENDAR.">=";

    function GET_MINFRAME_INTERVAL return LONG_DURATION
                              renames SMT_CALENDAR.GET_MINFRAME_INTERVAL;

TIME_INITIAL_VALUE : constant TIME := TIME_OF(1994, 1, 1, 0.0); ––1.1.1994 why not

––––––––––––––––––––––––––––––
–– STATE_CODE
––––––––––––––––––––––––––––––
type STATE_CODE is new DB_DEFINITIONS.STATE_CODE;
STATE_CODE_INITIAL_VALUE : constant STATE_CODE :=  (others => ' ');

––––––––––––––––––––––––––––––
–– PULSE
––––––––––––––––––––––––––––––
type PULSE is new BOOLEAN;
PULSE_INITIAL_VALUE : constant PULSE := TRUE;
PULSE_NOT_TRIGGERED : constant PULSE := FALSE;
PULSE_TRIGGERED     : constant PULSE := TRUE;
```

```
------------------------------
-- BURST_PULSE
------------------------------
type BURST_PULSE is new NUMERIC_TYPES.UNSIGNED_INTEGER32;
BURST_PULSE_INITIAL_VALUE : constant BURST_PULSE := 0;
BURST_PULSE_NOT_TRIGGERED : constant BURST_PULSE := 0;

------------------------------
-- BYTE_STREAM
------------------------------
type BYTE_STREAM (MAX_LENGTH : UNSIGNED_INTEGER) is
record
   ACTUAL  : UNSIGNED_INTEGER;
   ELEMENTS: UNSIGNED_BYTE_VECTOR (1 .. MAX_LENGTH);
end record;

type BYTE_STREAM_ACCESS is access BYTE_STREAM;
-- several functions concerning BYTE_STREAM

BYTE_STREAM_INITIAL_VALUE : constant BYTE_STREAM_ACCESS := null;

  type IO_VALUE_TYPE (SW_TYPE : CSS_SW_TYPES := NONE) is
  record
     case SW_TYPE is
       when NONE => null;
       when UNSIGNED_BYTE_TYPE =>
              UNSIGNED_BYTE_VALUE: UNSIGNED_BYTE := UNSIGNED_BYTE_INITIAL_VALUE;
       when SIGNED_BYTE_TYPE =>
              SIGNED_BYTE_VALUE:  SIGNED_BYTE := SIGNED_BYTE_INITIAL_VALUE;
       when UNSIGNED_SHORT_WORD_TYPE => UNSIGNED_SHORT_WORD_VALUE:
              UNSIGNED_SHORT_WORD := UNSIGNED_SHORT_WORD_INITIAL_VALUE;
       when SIGNED_SHORT_WORD_TYPE => SIGNED_SHORT_WORD_VALUE:
              SIGNED_SHORT_WORD := SIGNED_SHORT_WORD_INITIAL_VALUE;
       when UNSIGNED_INTEGER_TYPE => UNSIGNED_INTEGER_VALUE:
              UNSIGNED_INTEGER := UNSIGNED_INTEGER_INITIAL_VALUE;
       when SIGNED_INTEGER_TYPE =>
              SIGNED_INTEGER_VALUE: SIGNED_INTEGER := SIGNED_INTEGER_INITIAL_VALUE;
       when LONG_REAL_TYPE =>
              LONG_REAL_VALUE: LONG_REAL := LONG_REAL_INITIAL_VALUE;
       when COMPLEX_TYPE =>
              COMPLEX_VALUE: COMPLEX := COMPLEX_INITIAL_VALUE;
       when BOOLEAN_TYPE =>
              BOOLEAN_VALUE: BOOLEAN := BOOLEAN_INITIAL_VALUE;
       when TIME_TYPE =>
              TIME_VALUE: TIME := TIME_INITIAL_VALUE;
       when LONG_DURATION_TYPE =>
              LONG_DURATION_VALUE: LONG_DURATION := LONG_DURATION_INITIAL_VALUE;
       when UNSIGNED_BYTE_VECTOR_TYPE =>
              UNSIGNED_BYTE_VECTOR_VALUE: UNSIGNED_BYTE_VECTOR_ACCESS := null;
       when SIGNED_BYTE_VECTOR_TYPE => SIGNED_BYTE_VECTOR_VALUE:
              SIGNED_BYTE_VECTOR_ACCESS := null;
       when UNSIGNED_SHORT_WORD_VECTOR_TYPE => UNSIGNED_SHORT_WORD_VECTOR_VALUE:
              UNSIGNED_SHORT_WORD_VECTOR_ACCESS := null;
```

```
      when SIGNED_SHORT_WORD_VECTOR_TYPE => SIGNED_SHORT_WORD_VECTOR_VALUE:
             SIGNED_SHORT_WORD_VECTOR_ACCESS := null;
      when UNSIGNED_INTEGER_VECTOR_TYPE => UNSIGNED_INTEGER_VECTOR_VALUE:
             UNSIGNED_INTEGER_VECTOR_ACCESS := null;
      when SIGNED_INTEGER_VECTOR_TYPE => SIGNED_INTEGER_VECTOR_VALUE:
             SIGNED_INTEGER_VECTOR_ACCESS := null;
      when REAL_VECTOR_TYPE => REAL_VECTOR_VALUE : REAL_VECTOR_ACCESS := null;
      when LONG_REAL_VECTOR_TYPE => LONG_REAL_VECTOR_VALUE:
             LONG_REAL_VECTOR_ACCESS := null;
      when COMPLEX_VECTOR_TYPE => COMPLEX_VECTOR_VALUE:
             COMPLEX_VECTOR_ACCESS := null;
      when BOOLEAN_VECTOR_TYPE => BOOLEAN_VECTOR_VALUE:
             BOOLEAN_VECTOR_ACCESS := null;
      when UNSIGNED_BYTE_MATRIX_TYPE =>
             UNSIGNED_BYTE_MATRIX_VALUE: UNSIGNED_BYTE_MATRIX_ACCESS := null;
      when SIGNED_BYTE_MATRIX_TYPE => SIGNED_BYTE_MATRIX_VALUE:
             SIGNED_BYTE_MATRIX_ACCESS := null;
      when UNSIGNED_SHORT_WORD_MATRIX_TYPE => UNSIGNED_SHORT_WORD_MATRIX_VALUE:
             UNSIGNED_SHORT_WORD_MATRIX_ACCESS := null;
      when SIGNED_SHORT_WORD_MATRIX_TYPE => SIGNED_SHORT_WORD_MATRIX_VALUE:
             SIGNED_SHORT_WORD_MATRIX_ACCESS := null;
      when UNSIGNED_INTEGER_MATRIX_TYPE => UNSIGNED_INTEGER_MATRIX_VALUE:
             UNSIGNED_INTEGER_MATRIX_ACCESS := null;
      when SIGNED_INTEGER_MATRIX_TYPE => SIGNED_INTEGER_MATRIX_VALUE:
             SIGNED_INTEGER_MATRIX_ACCESS := null;
      when REAL_MATRIX_TYPE =>
             REAL_MATRIX_VALUE : REAL_MATRIX_ACCESS := null;
      when LONG_REAL_MATRIX_TYPE => LONG_REAL_MATRIX_VALUE:
             LONG_REAL_MATRIX_ACCESS := null;
      when COMPLEX_MATRIX_TYPE => COMPLEX_MATRIX_VALUE:
             COMPLEX_MATRIX_ACCESS := null;
      when BOOLEAN_MATRIX_TYPE => BOOLEAN_MATRIX_VALUE:
             BOOLEAN_MATRIX_ACCESS := null;
      when REAL_TYPE =>
             REAL_VALUE : REAL := REAL_INITIAL_VALUE;
      when STATE_CODE_TYPE =>
             STATE_CODE_VALUE: STATE_CODE := STATE_CODE_INITIAL_VALUE;
      when PULSE_TYPE =>
             PULSE_VALUE: PULSE := PULSE_INITIAL_VALUE;
      when BURST_PULSE_TYPE =>
             BURST_PULSE_VALUE : BURST_PULSE := BURST_PULSE_INITIAL_VALUE;
      when BYTE_STREAM_TYPE =>
             BYTE_STREAM_VALUE : BYTE_STREAM_ACCESS := null;
      when LAST_TYPE => null;
    end case;
end record;
```

```
type IO_SCALAR_VALUE_TYPE (SW_TYPE : CSS_SW_TYPES := NONE) is
record
   case SW_TYPE is
     when NONE => null;
     when UNSIGNED_BYTE_TYPE =>
          UNSIGNED_BYTE_VALUE: UNSIGNED_BYTE := UNSIGNED_BYTE_INITIAL_VALUE;
     when SIGNED_BYTE_TYPE =>
          SIGNED_BYTE_VALUE:  SIGNED_BYTE := SIGNED_BYTE_INITIAL_VALUE;
     when UNSIGNED_SHORT_WORD_TYPE => UNSIGNED_SHORT_WORD_VALUE:
          UNSIGNED_SHORT_WORD := UNSIGNED_SHORT_WORD_INITIAL_VALUE;
     when SIGNED_SHORT_WORD_TYPE => SIGNED_SHORT_WORD_VALUE:
          SIGNED_SHORT_WORD := SIGNED_SHORT_WORD_INITIAL_VALUE;
     when UNSIGNED_INTEGER_TYPE => UNSIGNED_INTEGER_VALUE:
        UNSIGNED_INTEGER := UNSIGNED_INTEGER_INITIAL_VALUE;
     when SIGNED_INTEGER_TYPE =>
          SIGNED_INTEGER_VALUE: SIGNED_INTEGER := SIGNED_INTEGER_INITIAL_VALUE;
     when LONG_REAL_TYPE =>
          LONG_REAL_VALUE: LONG_REAL := LONG_REAL_INITIAL_VALUE;
     when COMPLEX_TYPE =>
          COMPLEX_VALUE: COMPLEX := COMPLEX_INITIAL_VALUE;
     when BOOLEAN_TYPE =>
          BOOLEAN_VALUE: BOOLEAN := BOOLEAN_INITIAL_VALUE;
     when TIME_TYPE =>
          TIME_VALUE: TIME := TIME_INITIAL_VALUE;
     when LONG_DURATION_TYPE =>
          LONG_DURATION_VALUE: LONG_DURATION := LONG_DURATION_INITIAL_VALUE;
     when REAL_TYPE =>
          REAL_VALUE : REAL := REAL_INITIAL_VALUE;
     when STATE_CODE_TYPE =>
          STATE_CODE_VALUE: STATE_CODE := STATE_CODE_INITIAL_VALUE;
     when PULSE_TYPE =>
          PULSE_VALUE: PULSE := PULSE_INITIAL_VALUE;
     when BURST_PULSE_TYPE =>
          BURST_PULSE_VALUE : BURST_PULSE := BURST_PULSE_INITIAL_VALUE;
     when others => null;
   end case;
end record;
```

```
   type IO_COMPOSITE_VALUE_TYPE (SW_TYPE : CSS_SW_TYPES := NONE) is
   record
      case SW_TYPE is
        when UNSIGNED_BYTE_VECTOR_TYPE =>
              UNSIGNED_BYTE_VECTOR_VALUE: UNSIGNED_BYTE_VECTOR_ACCESS := null;
        when SIGNED_BYTE_VECTOR_TYPE => SIGNED_BYTE_VECTOR_VALUE:
                      SIGNED_BYTE_VECTOR_ACCESS := null;
        when UNSIGNED_SHORT_WORD_VECTOR_TYPE => UNSIGNED_SHORT_WORD_VECTOR_VALUE:
                      UNSIGNED_SHORT_WORD_VECTOR_ACCESS := null;
        when SIGNED_SHORT_WORD_VECTOR_TYPE => SIGNED_SHORT_WORD_VECTOR_VALUE:
                     SIGNED_SHORT_WORD_VECTOR_ACCESS := null;
        when UNSIGNED_INTEGER_VECTOR_TYPE => UNSIGNED_INTEGER_VECTOR_VALUE:
                     UNSIGNED_INTEGER_VECTOR_ACCESS := null;
        when SIGNED_INTEGER_VECTOR_TYPE => SIGNED_INTEGER_VECTOR_VALUE:
                      SIGNED_INTEGER_VECTOR_ACCESS := null;
        when REAL_VECTOR_TYPE => REAL_VECTOR_VALUE : REAL_VECTOR_ACCESS := null;
        when LONG_REAL_VECTOR_TYPE => LONG_REAL_VECTOR_VALUE:
                      LONG_REAL_VECTOR_ACCESS := null;
        when COMPLEX_VECTOR_TYPE => COMPLEX_VECTOR_VALUE:
                    COMPLEX_VECTOR_ACCESS := null;
        when BOOLEAN_VECTOR_TYPE => BOOLEAN_VECTOR_VALUE:
                    BOOLEAN_VECTOR_ACCESS := null;
        when UNSIGNED_BYTE_MATRIX_TYPE =>
              UNSIGNED_BYTE_MATRIX_VALUE: UNSIGNED_BYTE_MATRIX_ACCESS := null;
        when SIGNED_BYTE_MATRIX_TYPE => SIGNED_BYTE_MATRIX_VALUE:
                      SIGNED_BYTE_MATRIX_ACCESS := null;
        when UNSIGNED_SHORT_WORD_MATRIX_TYPE => UNSIGNED_SHORT_WORD_MATRIX_VALUE:
                      UNSIGNED_SHORT_WORD_MATRIX_ACCESS := null;
        when SIGNED_SHORT_WORD_MATRIX_TYPE => SIGNED_SHORT_WORD_MATRIX_VALUE:
                    SIGNED_SHORT_WORD_MATRIX_ACCESS := null;
        when UNSIGNED_INTEGER_MATRIX_TYPE => UNSIGNED_INTEGER_MATRIX_VALUE:
                     UNSIGNED_INTEGER_MATRIX_ACCESS := null;
        when SIGNED_INTEGER_MATRIX_TYPE => SIGNED_INTEGER_MATRIX_VALUE:
                      SIGNED_INTEGER_MATRIX_ACCESS := null;
        when REAL_MATRIX_TYPE => REAL_MATRIX_VALUE : REAL_MATRIX_ACCESS := null;
        when LONG_REAL_MATRIX_TYPE => LONG_REAL_MATRIX_VALUE:
                      LONG_REAL_MATRIX_ACCESS := null;
        when COMPLEX_MATRIX_TYPE => COMPLEX_MATRIX_VALUE:
                    COMPLEX_MATRIX_ACCESS := null;
        when BOOLEAN_MATRIX_TYPE => BOOLEAN_MATRIX_VALUE:
                    BOOLEAN_MATRIX_ACCESS := null;
        when BYTE_STREAM_TYPE =>
               BYTE_STREAM_VALUE : BYTE_STREAM_ACCESS := null;
        when others => null;
      end case;
   end record;

 end CSS_TYPES;
```

7.5 Model Observation & Control

7.5.1 Basics

The **M**odel **O**bservation and **C**ontrol **S**ystem (MOCS) has to accomplish two main task:

* CSS system commanding and
* CSS system state display

CSS system commanding includes the commanding of the Kernel (the simulator) and the definition of the system presentation (on–line definition for monitoring, logging and tracing).

CSS system observation functions provide the display of the states of e.g. the MOCS users, MOCS itself, the Kernel, the **I**mmediate **C**ommand **P**rocessor (ICP) and the monitored values.

CSS distinguishes two execution phases: the development phase and the operational phase.

Mocs can be started:

* by I_MDB during the ***development phase*** (this is described later in this document)
  Commands can be entered graphically from a MOCS main window or additional monitoring windows.
* by VICOS/HCI–ICP during the ***operational phase*** (MOCS starts up in real time mode and automatically accepts connections with ICP).
  Commands can be entered as a textual command string in a HLCL command window which is controlled either locally by ICP SW or remotely by VICOS SW. (The available commands are part of the HLCL description.)

Note that this chapter concentrates on the graphical commanding of MOCS (the development phase). The commanding of CSS during the operational phase is not part of this chapter.

The following steps describe in a short form how to proceed to get a simulation running (during the development phase).
During model editing the user should perform preliminary tests to check wether the model works according to the implementation goals or not.

1. Select the desired model in the I_MDB window and choose **MOCS**.
2. **Start** or **Connect** the simulator in the Simulation Controller window (simulator subview).
3. Select the host machine for the kernel.
4. Select the simulation state and **Load** it in the Simulation Controller (simulation subview).
5. Push the **Open Observer** button in the Simulation Controller (simulator subview) and select a simulation table. Open the simulation table in the Session Observer window.
6. Push the **Set** simulation mode button in the Simulation Controller (simulation subview) and set the simulation parameters.
7. **Start** the simulation in the Simulation Controller (simulation subview).
8. Observe the simulation results and messages in the Session Observer window and the MOCS Console window.

9. Manipulate input variables according to the testing objective.
10. **Stop** the simulation in the Simulation Controller (simulation subview).
11. **Stop** or **Disconnect** the simulator in the Simulation Controller (simulator subview).
12. Quit the CSS model evaluation session.
13. Now the logging files are available for inspection.

Detailed informations about the individual steps are given in the next sections.

7.5.2 Starting MOCS from I_MDB

The CSS invocation function provided in I_MDB allows to start the CSS tool. This functions can ***only*** be called up when the user is within a ***CCU scope***.

After a model has been successfully configured you can start MOCS to execute the model.

A description how to navigate in the data base is given in the procedures in section 7.4.4.1. They describe in detail how to start I_MDB, how to select CCU and CDU and how to start CSS from I_MDB.

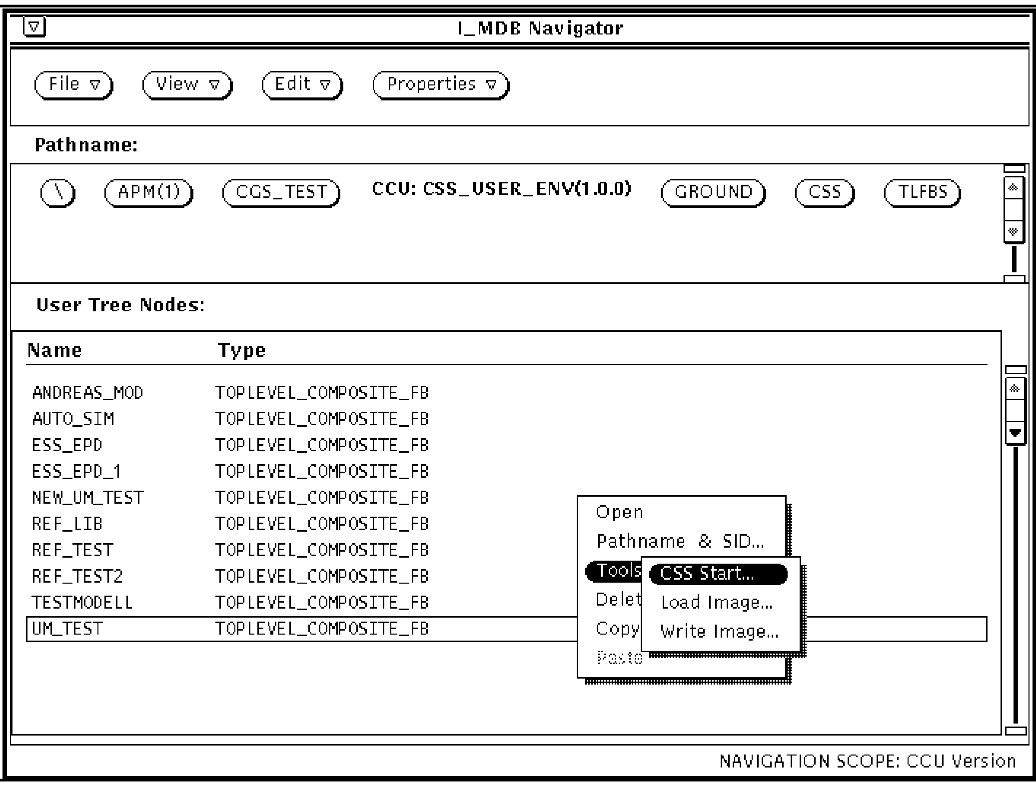


Figure 7–90 : *The CDU contains several models*

The preconditions for the following procedure are:

* a CCU is selected which contains all available references
* the CDU which contains the desired model is open

Start the CSS User Interface for Model execution

Start the CSS User Interface for Model execution

- **Select** the model in the I_MDB window. (see Figure 7–90)
- Press the right mouse button and select **Tools –> CSS Start...** from the pop–up menu.
- A confirmation window pops up. Press the **Ok** button.
- The CSS scope check window appears. MDE is pre–selected. Press the **MOCS** button. (see Figure 7–91)
- Press the **Start CSS...** button.
- The Database Browser window and the Simulation Controller window will be opened automatically.

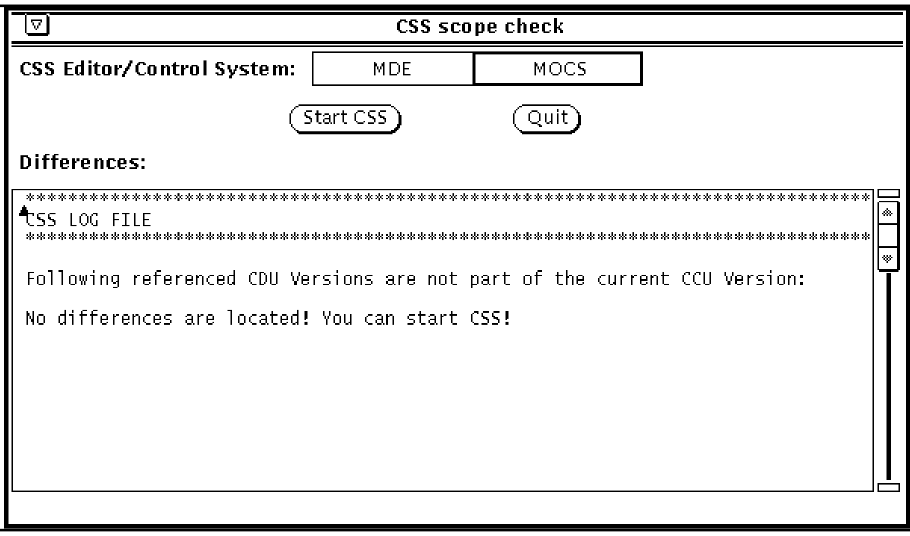


Figure 7–91 : *Select MOCS in the CSS scope window to start the model execution*

The CSS Simulation Controller window appears (additionally the DBB window pops up).

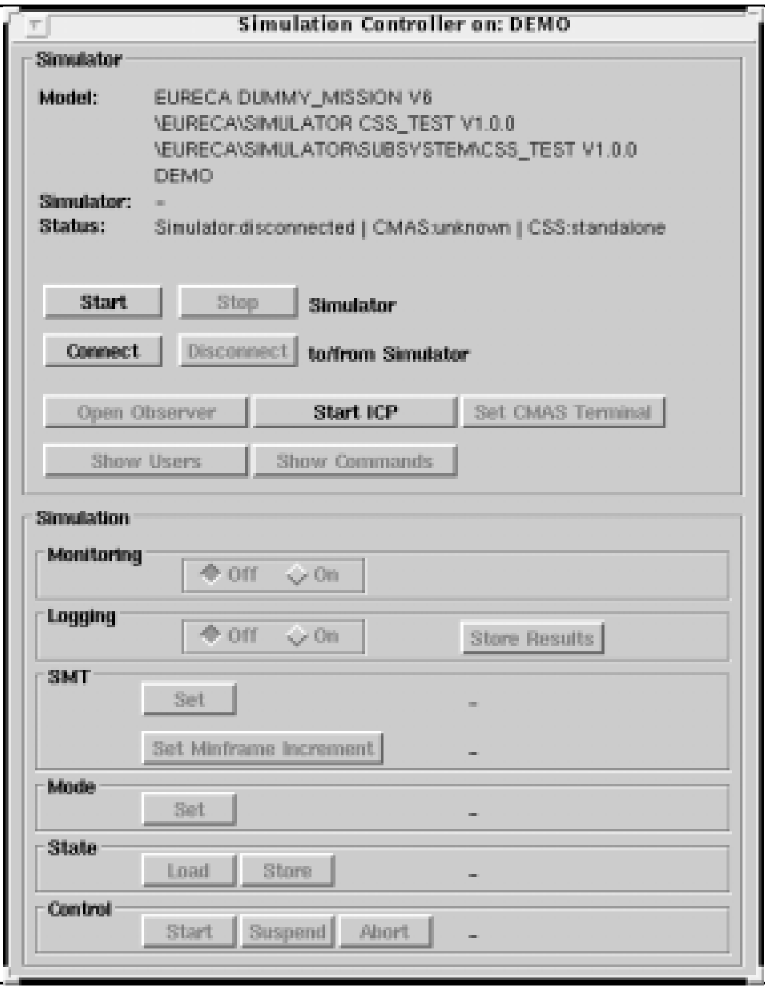


Figure 7–92 *: The initial CSS Simulation Controller window*

The window shows the following informations:

- * the **model identification**
  the element, mission identification, system_tree_version,
  the CCU with version, issue and revision,
  the CDU with version, issue and revision
  and the model name
- * the **simulation session status**,
  defaults are: **disconnected** for the simulator, **unknown** for CMAS and **standalone** for the CSS configuration (if CSS was called from I_MDB)

The CSS Simulation Controller window consists of two parts. The upper part (simulator subview) commands the start and stop of the kernel (i.e. the simulator), the connection/suspension to/from an already running simulator. Buttons give access to additional functions, e.g. the predefined simulation tables, a list of already running simulators. With the help of these function the user prepares his/her simulation environment.

The lower window part (simulation subview) contains the buttons to control the currently running simulation session. The user selects the simulation mode, i.e. stepwise or continuous model execution and uses buttons to control the simulation execution (start, suspend and abort).

7.5.3 Starting the simulator

The buttons in the upper part (simulator subview, see Figure Figure 7–92) have the following functions:

* the **Start** *Simulator* button
  starts the Kernel (simulator) on a machine of your choice.
* the **Stop** *Simulator* button
  stops the Kernel.
* the **Connect** *to/from Simulator* button
  connects the user to a pre–selected and already running Kernel (simulator).
* the **Disconnect** *to/from Simulator* button
  disconnects the user from the simulator. The Kernel stays active.
* the **Open Observer** button
  opens a separate window where you can load and activate a simulation table and make on–line changes
* the **Show Users** button
  displays a list of all users connected to a simulator and is the place, where user privileges are requested and withdrawn
* the **Show Commands** button
  shows all pending commands in a separate window (see Figure 7–94)

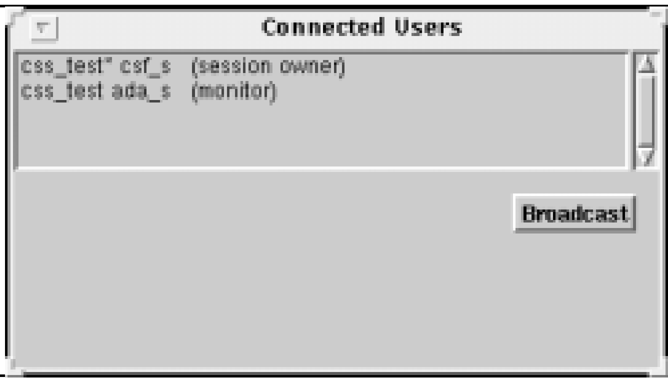


Figure 7–93 : *The Connected Users window shows a list of connected users and the status*

* the **Set CMAS Terminal** button
  is normally disabled. It will be activate if CMAS is running. The button allows to switch of remote terminals.
* the **Start ICP** button
  opens the ICP input window and starts the ICP. The ICP (Immediate Command Processor) window serves as an interactive gateway to the CSS commanding via HLCL.

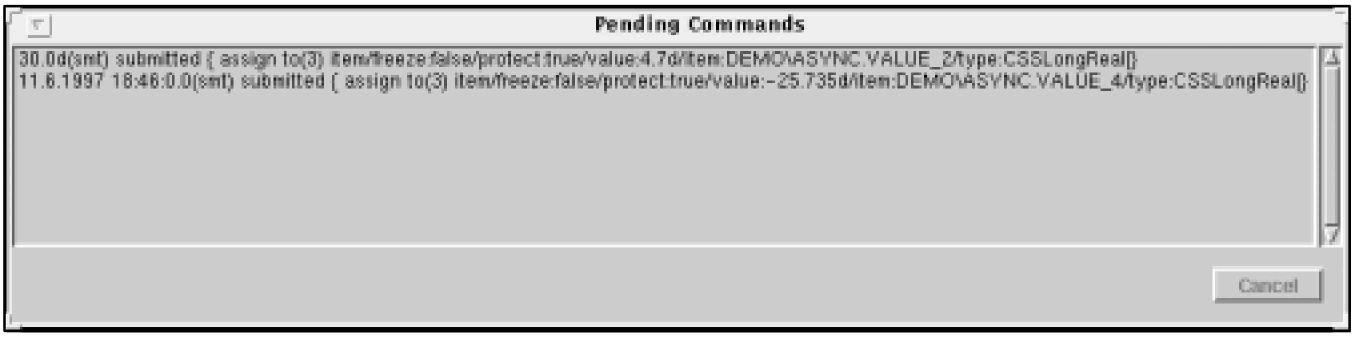


Figure 7–94 : *The pending commands window shows the commands and the affected model items*

Starting the Simulator

- Press the **Start** *Simulator* button using the left mouse button. The **Start Simulator** window pops up. (see Figure 7–95)
- Type the machine name in the **Kernel Host** field (see Figure 7–95). The actual name of the host machine depends on the HW environment where you are working.
- Press the **Apply** button. The status in the status line changes accordingly.

As long as the model itself is the item under test it's not necessary to start CMAS. CMAS is needed as the connecting part to HW.

- Click on the *Yes* button of the **Start CMAS** box.

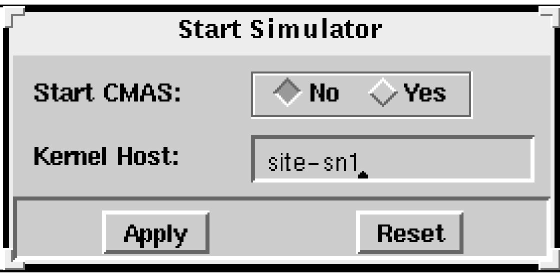


Figure 7–95 : *The Start Simulator window allows to specify the host machine*

Instead of starting the simulator every time you start a model execution it's possible to disconnect from the simulator and to re–connect at a later time. (This feature is only useful if you proceed with your work using the same model i.e. no changes were made). You leave the simulator using the **Disconnect** *to/from Simulator* button. The next time you can connect to the simulator which was still running in the meantime.

Connecting to a Simulator

Precondition for the connection to a simulator is that a session was quit with the **Disconnect** *to/from Simulator* button.

- Press the **Connect** *Simulator* button using the left mouse button. The **Simulators** window pops up.
- Select the simulator by clicking on the name in the list (see Figure 7–96). The name becomes highlighted.

Note that only those simulators are displayed which belong to the previously selected model.

- Press the **Apply** button. The status in the status line changes accordingly.

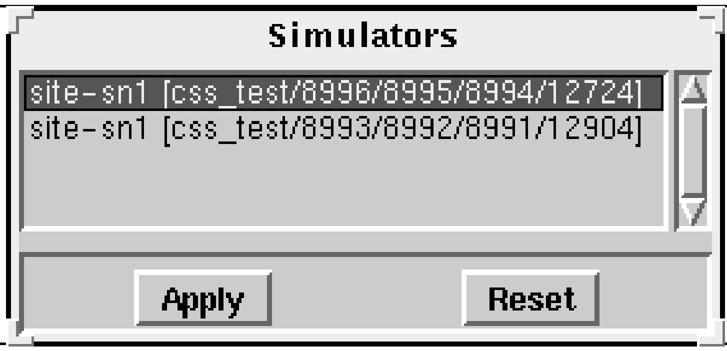


Figure 7–96 : *Select the kernel from the list of running simulators*

Note that the list of running simulators you see in the Simulators window belongs to just one model (the one you selected previously in I_MDB).
Make sure that you have finished your working task (i.e. the simulation is stopped) before working with another model.

7.5.4 User authorisation

The user who starts the simulator is called the ”session owner”. He/she is the person responsible for logging values, injecting values into the model, and finally stopping or restarting the simulator. All other users may connect to the session and may monitor values but are not allowed to control the simulation session.
The session owner can share his privileges with other users. As seen in Figure 7–97 the session owner is the user css_test working on the machine csf_s. A user working under the same user name but on a different workstation has the monitor privilege only.

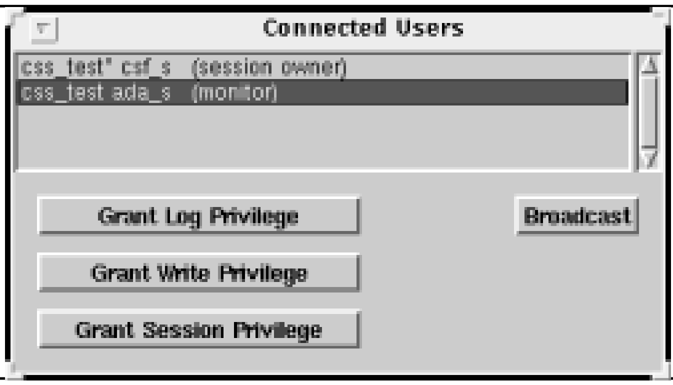


Figure 7–97 : *The selected user can receive privileges from the session owner*

To request a special privilege the user has to press the corresponding button. The message will be created automatically and send directly to the session owner (see Figure 7–98)

Figure 7–98 : *The user css_test on ada_s sends a request for the logging privilege*

If the session owner agrees to the request he/she selects the user in the Connected Users window (the user name must be highlighted) and pushes the **Grant Log Privilege** button. The entry in the Connected Users window changes accordingly (see Figure 7–99)

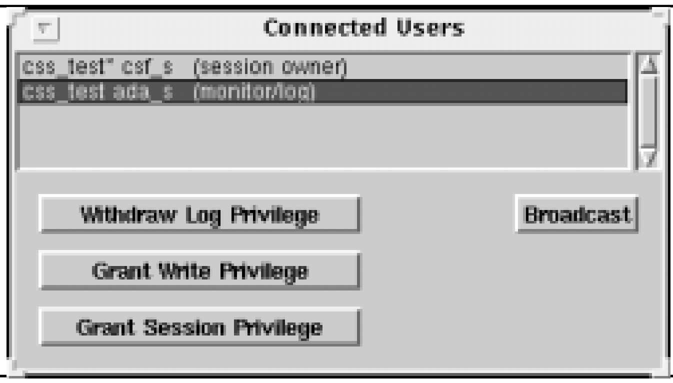


Figure 7–99 : *The user css_test on ada_s received the logging privilege.*

The session owner is allowed to withdraw given privileges.

Important: There is always only one session owner. As soon as the session owner gives the ***Session*** *privilege to another user he/she looses the session ownership. The user with the* ***Session*** *privilege is the session owner.*

The **broadcast** button allows to send a message to all connected users. Press the broadcast button and an input window appears.

Figure 7–100 : *The broadcast message is send to **all** connected users*

7.5.5 Starting a simulation session

Figure 7–101 : *The CSS Simulation Controller window with running Simulator*

The buttons in the lower part (simulation subview, see Figure Figure 7–101) have the following functions:

* the **Monitoring: On** and **Monitoring: Off** buttons
  switch monitoring on or off
* the **Logging: On** and **Logging: Off** buttons
  switch logging on or off
* the **Logging: Store Results** button
  allows to store the logging results (i.e. the data accumulated since the last storing or since the start of the simulation) in the file system (i.e. create archive files and optionally data set info files and data set files)
* the **SMT: Set** button
  allows to set the Simulated Mission Time (SMT). Per default the value is set to the machine time when the kernel starts.
* the **SMT: Set Minframe Increment** button
  allows to set the duration of a minframe in SMT.

* the **Mode: Set** button
  opens a separate window where you can adjust the parameters for model execution.
* the **State: Load** button
  opens a window with a list of available simulation states. The selected simulation state will be loaded.
* the **State: Store** button
  stores the current state of the simulation. In later sessions it is possible to start at that point again.
* the **Control: Start** button
  starts model execution.
* the **Control: Suspend** button
  halts model execution if you are running the model in continuous mode.
* the **Control: Abort** button
  stops model execution if the simulation is running faulty, i.e. an error condition occurred during simulation execution.

The monitoring and logging functions are performed as soon as the respective **On** button is selected and the simulation is started.

Note that there are no ***On/Off*** *buttons for tracing definitions. The tracing function is active as soon as the simulation is started.*

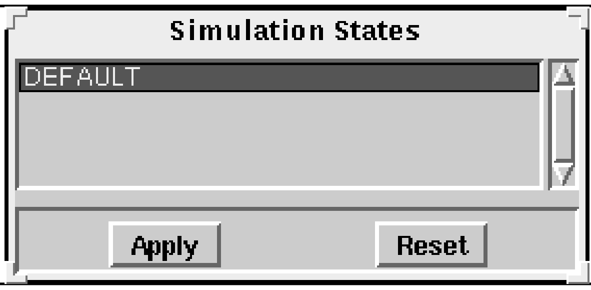


Figure 7–102 : *Select the simulation state from a list*

Starting a simulation session for model testing

- Press the **Load** S*tate* button using the left mouse button. The **Simulation States** window pops up.
- Select a simulation state by clicking on the name in the list (see Figure 7–102). The name becomes highlighted.

The simulation will start execution in the state you just loaded. The DEFAULT state is the initial state for the model.

- Press the **Apply** button. The status in the status line changes accordingly.
- Press the **Set** *Mode* button. The **Simulation Mode** window appears. (see Figure 7–103)

For first tests ***step mode*** *is recommended.*

- Click in the **Stepwise** box.
- Press the **Apply** button.
- Press the **Set Minframe Increment** *SMT* button. An input window appears.
- Click in the **Unit** box (select either **Seconds**, **Minutes** or **Hours**)
- Type the desired duration in the **Value** input field.
- Press the **Apply** button.
- Click on the **Monitoring On** button. The button becomes highlighted.
- Click on the **Logging On** button. The button becomes highlighted.
- Press the **Open Observer** button. The **Simulation Tables** window pops up.
- Select a simulation table from the list by clicking on its name in the list. (see Figure 7–104).
- Press the **Apply** button. *to be continued.......*

- Resize the session observer window conveniently.
- Press the **Start** *Control* button.

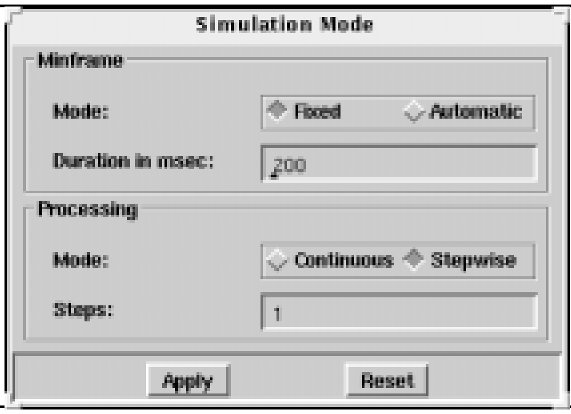


Figure 7–103 : *Setting the simulation mode parameters*

A more detailed description of the different execution modes and the timing capabilities will be given in the following sections.

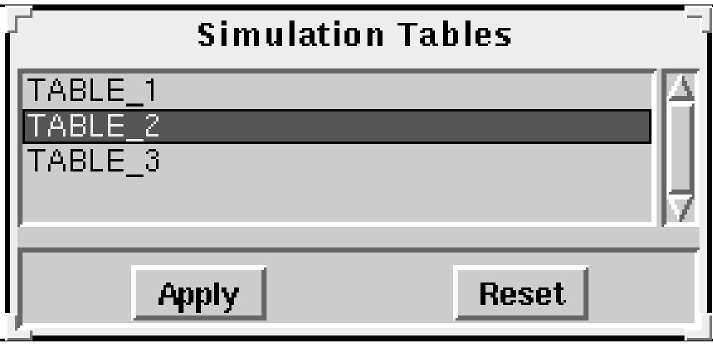


Figure 7–104 : *Select a simulation table from the list of available tables*

An overview about all testing capabilities (i.e. error injection into the model, parameter checks) will be given in the following sections.

Closing a simulation session

- If the simulation is running in continuous mode, press the **Suspend** *Control* button.
- If monitoring is enabled, click on the **Monitoring Off** button. The button is highlighted.
- If logging is enabled, click on the **Logging Off** button. The button is highlighted.

Perform the following steps if there are open simulation tables.

- Press the S**tore Table** button to save changes in the table you made during the simulation session. An input window appears.
- Replace the table name by a new one if you want.
- Press the **Apply** button to save the changed table.

- Move the mouse pointer in the label of the Session Observer window and select **Quit** from the window pop–up menu. The Session Observer window disappears.
- Move the mouse pointer into the Simulation Controller window and press the **Stop** *Simulator* button.
- Move the mouse pointer in the label of the Simulation Controller window and select **Quit** from the window pop–up menu. The Simulation Controller window disappears.

7.5.6 Simulation execution

7.5.6.1 The Session Observer window

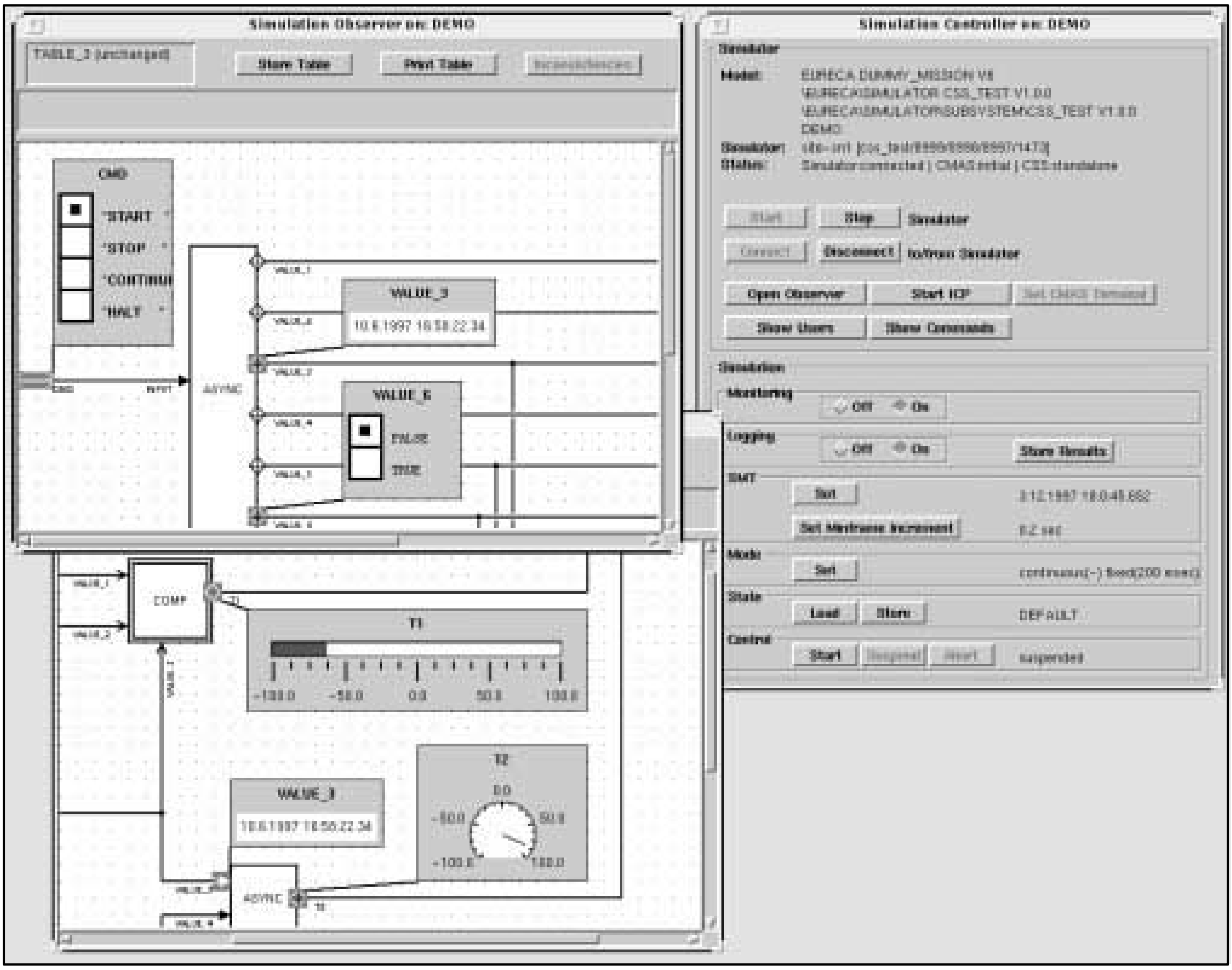


Figure 7–105 : *The Simulation Controller window with two active Session Observer windows showing different levels of the model*

7.5.6.2 Basics

The main purpose of the Model Observation and Control function of CSS is to serve as the user interface for interactive simulation control and monitoring. This is done via the **Session Observer** window. After loading a simulation state you can press the **Open Observer** button. The **Simulation Tables** window allows the selection of the desired simulation table from a list.

Note that the precondition for on–line loading and changing tables is that a simulation state is loaded. Otherwise the command button **Open Observer** *is disabled.*

It is possible to load more than one simulation table and to switch during simulation execution to another table (see Figure 7–105), tables can be changed on–line and the changes can be stored.

The three buttons in the upper part (see Figure 7–106) have the following functions:

* the **Store Table** button
  stores the changes made during on–line table editing
  (either the table is updated or a new table can be created)
* the **Print Table** button
  prints the parameters and their monitoring, tracing or logging status
* the **Inconsistencies** button
  shows the inconsistent definitions with respect to the current simulation model contained in the simulation table

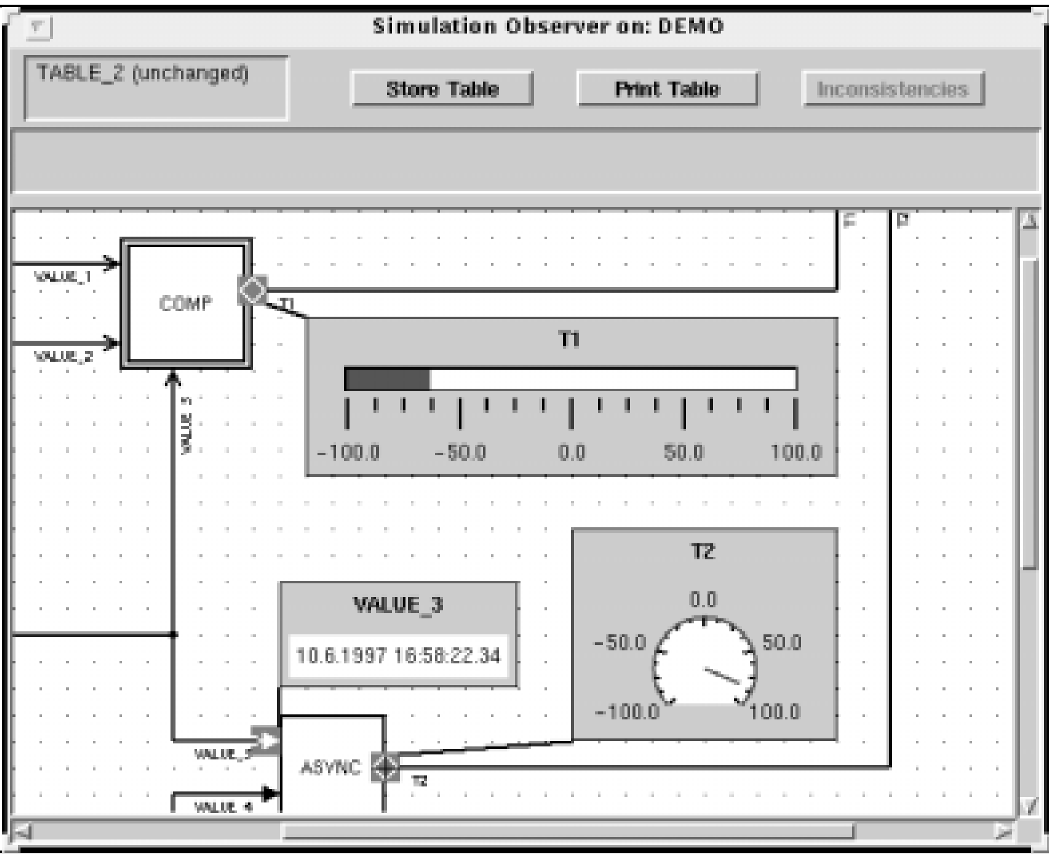


Figure 7–106 : *The Session Observer used to oberve the simulation*

7.5.6.3 On–line monitoring

All functions already described in section 7.4.7.9 are available during on–line monitoring, including the creation of new simulation tables. This can be achieved by editing an already existing simulation table and storing it with a different – currently unused – name.

A set of additional functions for on–line testing is provided

* **snapshot –> to screen**
  Display the current values of the selected atomic outputs and inputs, parameters and model inputs and outputs in the MOCS Console window.

* **snapshot all –> to screen**
  Display the current values of all atomic outputs, parameters and model inputs in the MOCS Console window.

* **snapshot –> to log file**
  Write the current values of the selected atomic outputs and inputs, parameters and model inputs and outputs to an archive file (refer to section 7.4.7.9.4 for more information on logging and archive files).

* **snapshot all –> to log file**
  Write the current values of all atomic outputs, parameters and model inputs to an archive file (refer to section 7.4.7.9.4 for more information on logging and archive files).

* **snapshot –> as HLCL sequence**
  Generate a file containing HLCL statements that assign the respective current value to the selected atomic outputs and inputs, parameters and model inputs and outputs. The MOCS Console window gives the name of the file generated.

* **snapshot all –> as HLCL sequence**
  Generate a file containing HLCL statements that assign the respective current value to all atomic outputs, parameters and model inputs. The MOCS Console window gives the name of the file generated.

The HLCL sequence is stored in the Unix file system in a directory which is determined by the environment variable CSS_LOG_DIR . The file is marked with the extension .SNAPSHOT_HLCL_#, the sharp sign indicates the serial number. The part of the name preceeding the extension matches the same pattern as for log and archive files (refer to section 7.4.7.9.4 for more information on names of log and archive files).
In contrast to the simulation model's simulation states which are automatically deleted when a new simulator kernel is configured, these HLCL sequences may survive such configurations and then subsequently be used to set the values of the model variables according to a previous state (i.e. the state at snapshot time). However, the user is responsible that the HLCL sequence is still compatible to the simulation model with respect to pathnames and types of the interface items assigned. If necessary, a HLCL sequence may be edited using a text editor to adapt it to a modified simulation model.

* **freeze**
  Freeze the last value of an atomic output or model input. The value will not change during further execution.

* **reactivate**
  Reactivate a frozen atomic output or model input.

* **assign –> don't freeze**
  Assign a value to an atomic output (which is marked with a 'write access' marker) or model input. The value will be sent once.

* **assign –> freeze**
  Assign a value to an atomic output (which is marked with a 'write access' marker) or model input, then freeze the atomic output or model input.

Note that you are not allowed to assign values to atomic inputs or model outputs.

Taking a snapshot to screen

- Select the item (input or output) in the Session Observer window by clicking on it with the left mouse button.

It is possible to get more than one value with one snap shot command.

- Select the desired input/output by clicking on it with the middle mouse button. Repeat this until all desired items are selected.
- Press and hold the right mouse button. Select the command **snapshot–>to screen** from the pop–up menu.
  The values appear immediately in the MOCS Console window (see Figure 7–107)

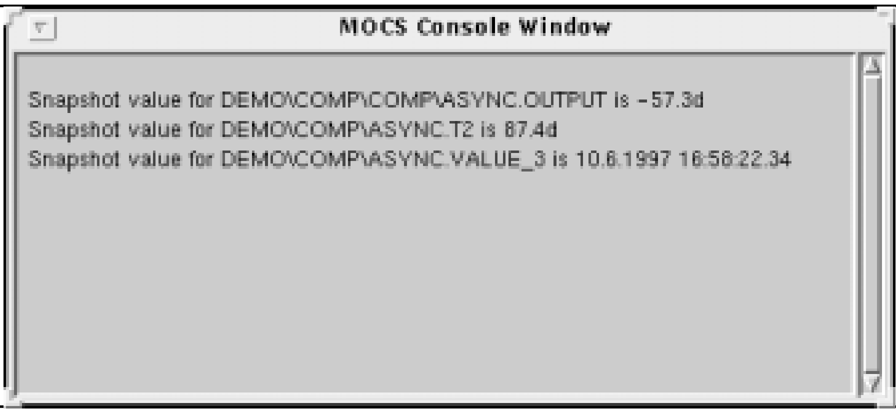


Figure 7–107 : *Snapshot values are displayed in the MOCS Console window*

Assign a value to an atomic output or model input

- Select the item by clicking on it with the left mouse button.
- Press and hold the right mouse button. Select **assign** from the pop–up menu. The Assign window appears (see Figure 7–108)
- Type the new value in the **Value** input field.

  Now you can freeze the item.

- Click into the **Freeze** box. The value will be valid for the item until the item is reactivated.
- Press the **Apply** button. The assign command will be performed immediately.

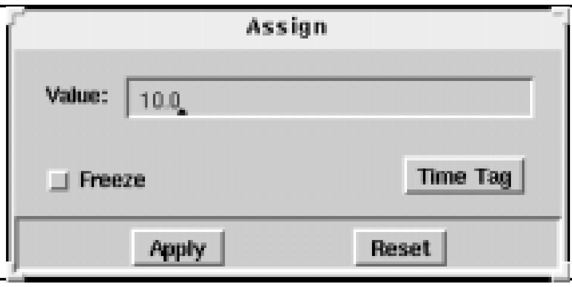


Figure 7–108 : *The Assign window*

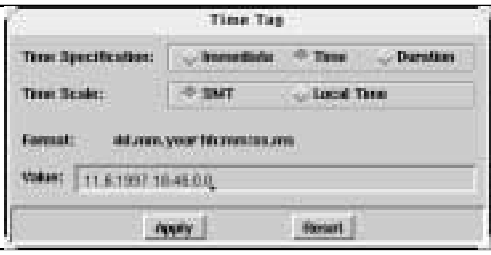


Figure 7–109 : *Specifying a time tag in SMT*

Assign a time tagged value to an atomic output or model input

- Select the item by clicking on it with the left mouse button.
- Press and hold the right mouse button. Select **assign** from the pop–up menu. The Assign window appears (see Figure 7–108)
- Type the new value in the **Value** input field.

  Now you can freeze the item.
- Click into the **Freeze** box. The value will be valid for the item until the item is reactivated.

  A time tag delays the execution of the assign command for a specified time.
- Press the **Time Tag** button. The Time Tag specification window appears (see Figure 7–109).
- Click into the **Time Specification: Time** box, then click either into the **Time Scale: SMT** or **Local Time** box.

- Type the date and time in the **Value** input field (see Figure 7–109)
  OR
- Click into the **Time Specification: Duration** box, then click either into the **Time Scale: SMT** or **Local Time** box.
- Click in the **Unit** box (select either **Seconds**, **Minutes** or **Hours**)
- Type the duration in the **Value** input field (see Figure 7–110)
- Press the **Apply** button. The assign command will be performed at the specified time or after a specified duration.
- A confirmation window pops up (see Figure 7–111). Press the OK button.

Figure 7–110 : *Specifying a duration in SMT*

Figure 7–111 :*The confirmation window certifies the submission of a time tagged assign command*

7.5.6.4 Time scales used during simulation execution

There are two independent time scaling systems used in the simulation:

* the SMT (Simulated Mission Time) constitutes the ”official” simulation time.
  The model developer can set the SMT increment per minframe according to his needs, so that the SMT clock can show the hours passing by (and not the actual local time used for model execution); an action which takes some hours in real time can be simulated in seconds.
* the LT (Local Time) is the actual time in which simulation execution takes place.
  In fixed minframe mode, the execution of each minframe lasts exactly the specified constant duration (the minimum value is 50 msec). In automatic minframe mode, each minframe lasts exactly the time needed for performing all computations; this results in varying minframe durations.

7.5.6.4.1 Setting the Simulated Mission Time

7.5.6.4.1.1 Setting the SMT starting point

As soon as you start the simulator, CSS sets the SMT to the current date and time which is provided by the kernel host.
You can choose any time you like as starting point.

Figure 7–112 : *The SMT input window*

The following input format is expected:

dd	day – one or two digits (1 – 31)
mm	month – one or two digits (1 – 12)
year	year – must be four digits
hh	hours – one or two digits (0 – 23)
mm	minutes – one or two digits (0–59)
ss	seconds – one or two digits (0–59)

ms milliseconds – up to three digits (0–999)

Day, month and year must be separated by a point; hours,minutes and seconds by a colon. Milliseconds must be separated by a point. Note that there is a blank between date and time.

Setting the SMT starting point

- If the simulation was already started, you have to load a simulation state first before you can change the time setting.
- Press the **Set** *SMT* button. The SMT input window pops up (see Figure 7–112).
- Type the desired date and time in the **Value** input field.
- Press the **Apply** button. The SMT is changed in the Simulation controller window.

7.5.6.4.1.2 Setting the SMT increment per minframe

The SMT minframe increment determines how long one minframe shall last in SMT.

Setting the SMT increment per minframe

- If the simulation was already started, you have to load a simulation state first before you can change the SMT increment per minframe.
- Press the **Set Minframe Increment** *SMT* button. The SMT increment input window pops up (see Figure 7–113).
- Click in the **Unit** box (select either **Seconds**, **Minutes** or **Hours**)
- Type the desired duration in the **Value** input field.
- Press the **Apply** button.

i – *i* *i* *i*

7.5.6.5 The non–Real Time Simulation Modes

Figure Figure 7–114 shows the default simulation mode setting suggesting the **Fixed** *Minframe* mode and the **Continuous** *Processing* mode.

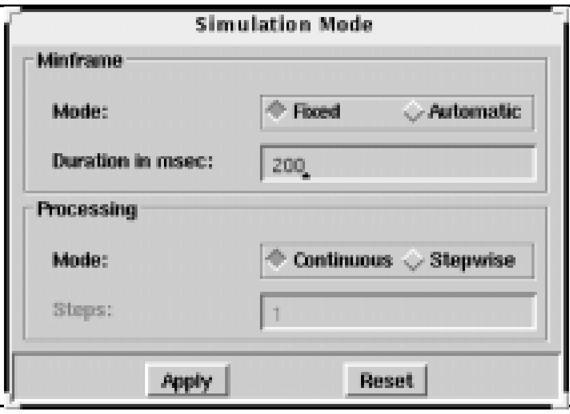


Figure 7–114 : *The Simulation Mode window in default state*

7.5.6.5.1 The Minframe Mode

In fixed minframe mode, the execution of each minframe lasts exactly a specific constant duration. You can specify this value in the **Duration in msec** input field (the minimum value is 50 msec).

The automatic minframe mode could also be called the ’as fast as possible’ mode. As soon as all computations for a step (minframe) are performed, the computations for the next step (minframe) are started. The minframe duration is determined by the amount of time needed to perform all calculations for a step and varies from step to step.

7.5.6.5.2 The Processing Mode

You can choose between either the continuous or the stepwise processing mode. For first checks stepwise processing mode is recommended. You can type the number of steps, i.e. minframes to be executed when you start the simulation, in the **Steps** input field. If you start the simulation by pressing the **Start** *Control* button in the Simulation Controller window, the simulation will automatically be suspended after the given number of steps are performed and you can inspect the changes of the model parameters.
In continuous processing mode the simulation will run until you press the **Suspend** *Control* button in the Simulation Controller window.

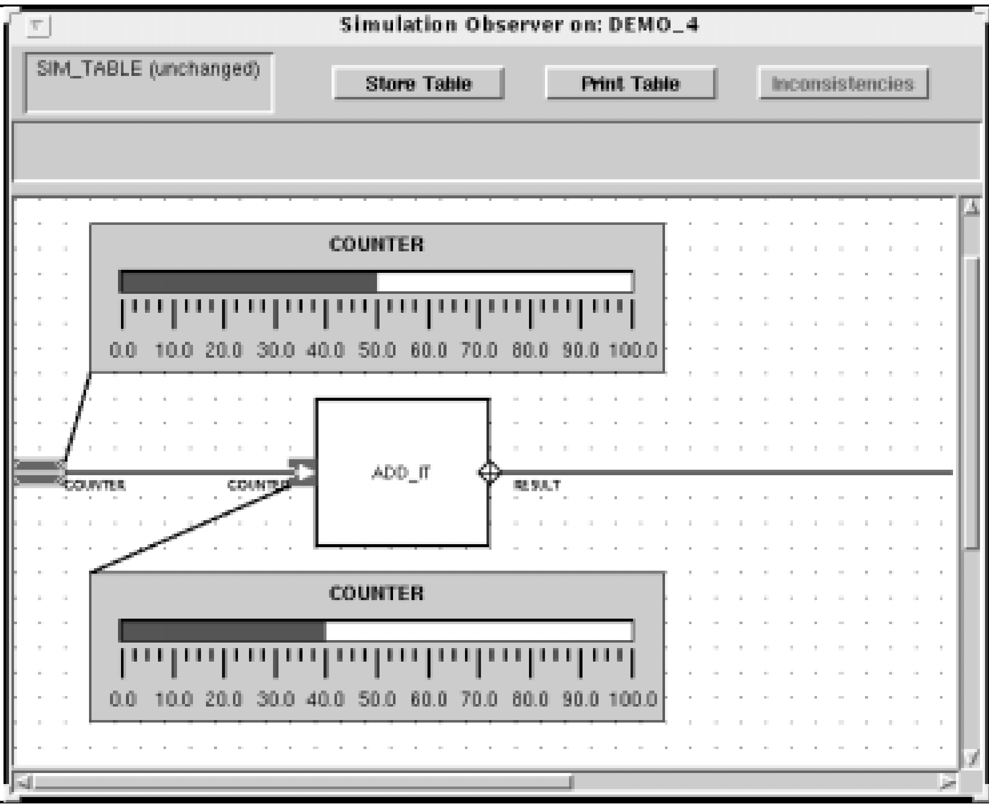


Figure 7–115 : *Monitoring the same parameter shows different values*

Figure 7–115 shows a puzzling situation. The picture shows the inside of a composite function block. The gauge element connected to COUNTER at the border of the composite displays the value 50. The monitoring element connected to the input of function block ADD_IT , COUNTER as well, shows the value 40. This situation comes up during stepwise processing mode (single step). All input buffers are written at the beginning of a time frame (this is the reason why the input connected monitoring element shows the value 40). Output buffers are written as soon as the function block has finished its calculations. (This is the reason why the other monitoring element, which is connected to an output on higher level, shows the value 50). You have to take into account such mechanisms to understand the monitoring capabilities.

7.5.6.6 The Real Time Simulation Mode

The **real time** mode will be automatically set as soon as the CSS is used in a configuration with hardware in the loop and started from VICOS. The SMT will be provided by the Master Timing Unit for all connected workstations.
The commanding is done via VICOS HCI which means that most of the CSS functions are disabled. Refer to section 7.5.7 for more information.

7.5.6.7 Creating new Simulation States

An initial simulation state was created during simulator kernel configuration. This simulation state sets the model to the start conditions.

Creating a new simulation state

- Take all actions to drive the model in a state you want to use as starting point for the next simulation session.
- Move the mouse pointer into the lower part of the Simulation Controller window (simulation subview) and press the **Store** *State* button. An input window pops up.
- Type a new name in the input field.
- Press the **Apply** button.

The simulation state defines the start–up state of your simulation. If you are not interested in starting the model just from the initial state, use MOCS to create more simulation states. Start the simulation and proceed execution until the desired state is reached, then store the simulation state. The next time you can set up the simulation in this state by loading the simulation state.

7.5.6.8 Errors during simulation execution

If an error occurs during simulation run the model status changes to Running Faulty.

Note that the simulation will be continued with the last valid parameters.

All simulation results achieved in this situation have to be treated as invalid. It is possible to store the simulation state for later error evaluation. The simulation has to be stopped with the **Abort** *Control* button. This ensures the proper end of the model execution.

7.5.7 Commanding CSS via HLCL primary commands

The ICP (Immediate Command Processor) window serves as an interactive gateway to the CSS commanding via HLCL.

General informations about the use of HLCL can be found in chapter H-8.2.

Activating the ICP window

- Select a model in I_MDB.
- Select from the pop–up menu **Tools–>Start CSS .....**
- A message window saying: *Tool has been started in batch mode* appears. Press the **Ok** button.
- The CSS scope check window appears. Select the **MOCS** button with the left mouse button then press the **Start CSS** button.
- The Simulation Controller window pops up. Press the **Start ICP** button in the window's upper part (simulator subview, see Figure 7–101). The MOCS Console window appears and the ICP window pops up.

The six buttons in the ICP window (see Figure 7–116) have the following functions:

* the **History** button
  opens a seperate history window which displays all commands raised during the current session
* the **Interrupt** button
  stops the execution of a command sequence.

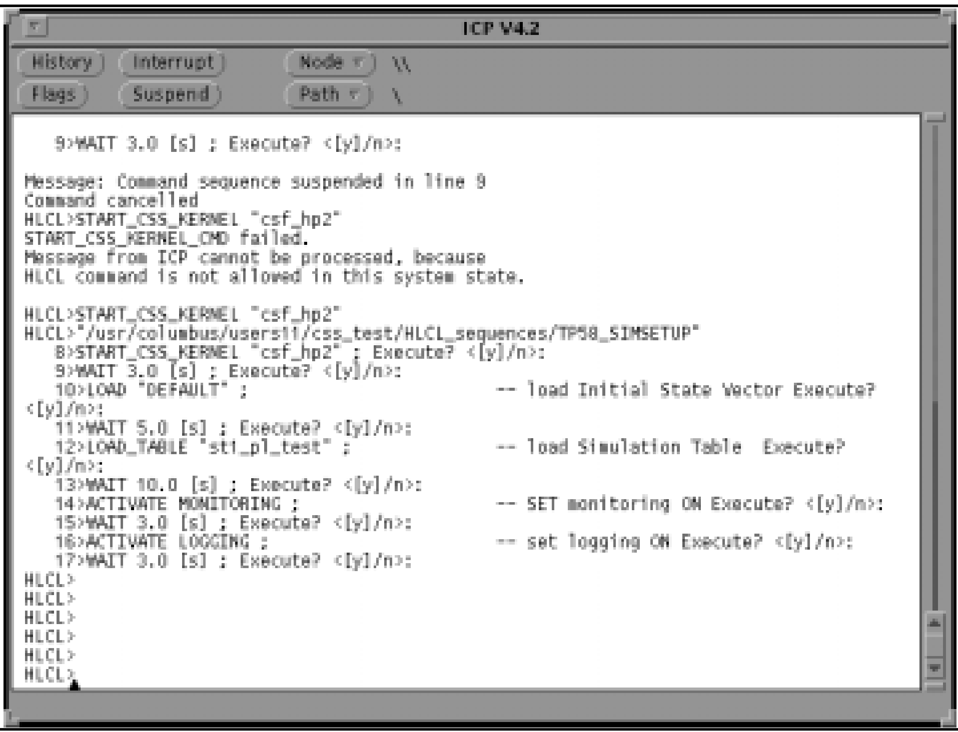


Figure 7–116 : *The ICP window is used for command sequence testing*

* the **Suspend** button
  suspends the execution of a command sequence
* the **Node** button
  allows a to change the default node. The *default node* is used to start APs and call library routines.
* the **Path** button
  allows a to change the default path. Wherever a path name is prefixed with \\, the prefix is expanded with the current value of the variable DEFAULT_PATH.
* the **Flags** button
  displays an interactive window (see Figure 7–117) and allows the setting of following predefined variables.

  ECHO:
  This variable defines whether commands executed from command sequences are echoed on the screen. If set to false, only start and stop of command sequences are reported. This variable is initially set to false.

  TRAP:
  This variable determines the error handling mechanism within command sequences. When set to true, an error interrupts the execution of the sequence and returns control to the session. The com mand sequence remains loaded and may be resumed with the RESUME command.

  STEP:
  This variable selects *single–step execution* of command sequences. When set to true, each command from a command sequence must be confirmed before execution.

KEEP:
When KEEP is set to true (and a log file is open), all interactively entered commands are logged in the currently open log file.

DEBUG:
This variable selects *debug mode*. When set to true (and a log file is open), all interactively typed commands are logged in expanded form in a commands log file.

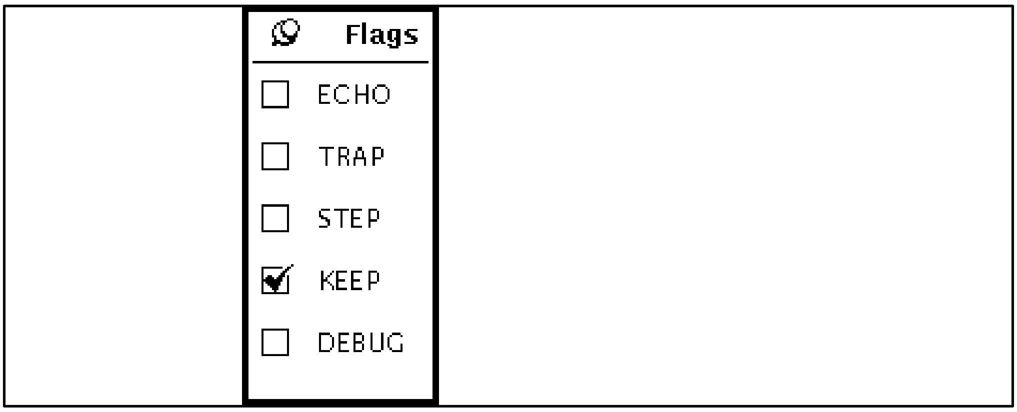


Figure 7–117 : *The flags can be set interactively*

?

On–line HLCL commanding

- An ICP window is opened, either fromthe CSS side or from the VICOS side.
- Press the **Flags** button and set the desired flags.
- Type the command and press the **Return** key on the keyboard.

Starting a HLCL command sequence

A command sequence is normally stored in the database. It is nevertheless possible that a command sequence is stored in the users own UNIX directory.

- Press the **Flags** button and select the flags ECHO, TRAP.
- *Note that if you select the STEP flag the user must acknowledge the execution of each step manually. Only for test purposes recommanded.*
- Enter the command sequence with its pathname and press the **Return** key on the keyboard.

7.5.8 Simulation results evaluation

After the simulation is finished and the kernel is stopped, the simulation results (i.e. logging and tracing tables) are available for further investigations in the Unix file system in a directory given by the environment variable CSS_LOG_DIR.
The results are stored in ASCII files under following names:
modelname_...._username_date_time.LOG_TXT and
modelname_...._username_date_time.ARCH_TXT

Note that files with the extension .LOG and .ARCH are incomplete binary files which result from a model execution abort. These files should be deleted.

In the .....ARCH_TXT you will find the function outputs & inputs logged and all external model inputs & outputs logged.

```
----------------------------------------------------
Log value:
Runtime ID: 7  SIGNED_INTEGER, Value:        93
Log value:
Runtime ID: 9  REAL, Value:-8.88000E+01
Log value:
Runtime ID: 12  REAL, Value:-8.45774E+01
 Local Time:            15/12/1995 13:36:34.652
 Simulated Mission Time: 15/12/1995 13:35:42.118
----------------------------------------------------
Log value:
Runtime ID: 7  SIGNED_INTEGER, Value:        93
Log value:
Runtime ID: 9  REAL, Value:-8.88000E+01
Log value:
Runtime ID: 12  REAL, Value:-8.25184E+01
 Local Time:            15/12/1995 13:36:34.852
 Simulated Mission Time: 15/12/1995 13:35:42.318
----------------------------------------------------
```

Figure 7–118 : *A part of the file .....ARCH_TXT*

The .....LOG_TXT file contains all system messages logged, all operator commands logged and all command response logged. Additionally the file lists the runtime IDs and the related model item paths.

The file ....LOG_TXT contains the list of runtime IDS which are indispensable to read the contents of the file ....ARCH_TXT.

```
*******************************************************************************
*                                                                             *
*                            CSS LOGGER BACKEND                               *
*                                                                             *
*                                 Version:                                    *
*                                                                             *
*                     CGS/CSS Version 4.2 from 01.12.95                       *
*                                                                             *
*                                                                             *
*                (C) 1995 DASA -- ERNO Raumfahrttechnik GmbH                  *
*                                                                             *
*******************************************************************************

Model Identification (based on Database):
  Element Configuration:´EURECA´
  Mission Name         :´DUMMY_MISSION´
  System Tree Version  : 4
  Library              :´\EURECA\SIMULATOR\TEST\MODELS_1´
  Version  : 1  Issue    : 0  Revision : 0
  Top Level FB Name    :´PL_TEST´
-----------------------------------------------------
Mapping (Runtime ID - Pathname):
 14   PL_TEST.PAYLOAD_ON
 15   PL_TEST.PAYLOAD_OFF
 7    PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.ERROR
 8    PL_TEST\PAYLOAD_NO_1\PL_MAX_TEMP.PL_MAX_TEMP
 9    PL_TEST\ENV_TEMP.VALUE
 10   PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.ACT_MFR_HEAT_CTR
 11   PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.ACT_MFR_RUDO_CTR
 12   PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.PAYLOAD_TEMP
 13   PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.PL_ON_OFF_STATE
 16   PL_TEST.PAYLOAD_TEMP
 17   PL_TEST.PL_ON_OFF_STATE
 3    PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.PL_MAX_TEMP
 4    PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.T_ENV
 5    PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.PL_OFF_CMD
 6    PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.PL_ON_CMD
 1    PL_TEST\PAYLOAD_NO_1\PL_MAX_TEMP
 2    PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL
-----------------------------------------------------
Command: LOGIN
```

Figure 7–119 : *The file ...LOG_TXT contains a list of runtime IDs*

7.6 ther Enditems

7.6.1 General

CGS provides a large number of predefined enditem types. A set of those types are used via specific tools, such as the FWDU, GWDU, MDE and CLS Editor/Compiler. The data for the remaining types are to be entered via the I_MDB and DDED tools. These are the types mainly defined for measurements and commands, and for the definition of the test configurations.

7.6.1.1 Enditem Editing via I_MDB and DDED

When starting the I_MDB tool (Task Selector –> Mission or Test Preparation), navigation in the MDB is provided. When navigated to an enditem, values for its aggregates can be entered using the ”Open” menu entry.

An alternative way to enter data interactively is by strating the Detailed Data Editor (DDED) for an item from within the I_MDB tool.

Refer to MDA User Manuals for detailed description.

7.6.2 End Item Types

7.6.2.1 Basic Types

The description of the masks and the detailed description of the data to be entered into the mask is not provided. However, Table 7–2 provides information about the value ranges of the MDB basic data types.

BASIC TYPE	VALUE RANGES
BASIS DATATYPES	
SINGLE_FLOAT	–3.40282E+38 to 3.40282E+38
DOUBLE_FLOAT	–1.79769313486231E+308 to 1.79769313486231E+308
INTEGER	– (2**31) to 2**31 –1
UNSIGNED_INTEGER	0 to 2**31 –1
BITSET	32 bits long, 1 or 0
HEXADECIMAL	255 bytes long, characters allowed ’0’ .. ’9’, ’A’ .. ’F’
PATHNAME	Syntax of an *MDB pathname and node name :* pathname ::= \node_name {\node_name } | \node_name ::= letter_or_digit_or_underline { letter_or_digit_or_underline } *with length of node_name <= 16* letter_or_digit_or_underline ::= letter | digit | underline In the MDB the node names are stored in capital letter.
STRING	– 0..MAX_STRING long, with MAX_STRING = 255.
LONG	– 0..MAX_STRING long, with MAX_STRING = 32000.
RAW	– 0..MAX_RAW long, with MAX_RAW = 255.

LONG_RAW	– 0..MAX_RAW long, with MAX_RAW = 32000.
DATE	DDMMYYHH24MISS
CGS ENUMERATIONS:	
CURVE_TYPE	POINT_PAIRS, LINEAR_POLYNOM, POLYNOM, IDENTICAL
BOOLEAN	TRUE, FALSE
CGS_NODE_PREFIX	HCI, CSS, TES, DBS
COMPARISON_OPERATOR	>, >=, <, <=, =, /=, in_range
CONDITION_TYPE	ENABLE_PROCESSING, SWITCH_LIMIT_SET, START_AP
COMPLETION_CODE	SUCCESS, FAILURE
EGSE_ADU_TYPE	STRUCTURED, UNSTRUCTURED, CCSDS PACKET
EGSE_CCSDS_TIME_ID	NO TIME FIELD, REALTIME COMMAND, TIME TAG, UNDEFINED
EGSE_PACKET_TYPE	CCSDS_DEFAULT_PACKET, CCSDS_MEMORY_DUMP_PACKET, CCSDS_DATA_SEGMENT_PACKET, CCSDS_ESSENTIAL_HK_PACKET, CCSDS_SYSTEM_HK_PACKET, CCSDS_PAYLOAD_HK_PACKET, CCSDS_SCIENCE_PACKET, CCSDS_SSMB_ANCILLIARY_PACKET, CCSDS_ESSENTIAL_COMMAND_PACKET, CCSDS_SYSTEM_COMMAND_PACKET, CCSDS_PAYLOAD_COMMAND_PACKET, CCSDS_MEMORY_LOAD_PACKET, CCSDS_RESPONSE_PACKET, CCSDS_REPORT_PACKET, CCSDS_EXCEPTION_PACKET, CCSDS_ACKNOWLEDGE_PACKET
EGSE_TC_TYPE	CCSDS_TC_PACKET
EGSE_BYTESTREAM_LAY-OUT_FORMAT	UNSIGNED_INTEGER, INTEGER, SINGLE_FLOAT, DOUBLE_FLOAT BINARY
EGSE_BITSTREAM_LAY-OUT_FORMAT	INTEGER, FLOAT, BINARY,UNSIGNED
EGSE_VALUE_TYPE	ASCII, HEX
EGSE_RAW_VALUE_TYPE	UNSIGNED_INTEGER, SIGNED_INTEGER, FLOAT, BYTESTREAM
EGSE_NODE_TYPE	WORKSTATION, TEST_NODE, SIMULATOR, DATABASE SERVER, FRONT–END EQUIPMENT, UNIT UNDER TEST
EGSE_SOFTWARE_TYPE	SAS —SAS executable EXECUTABLE—any FEE or UUT executable DATA_FILE —any SAS, FEE or UUT data file
EGSE_EXECUTION_MODE	NORMAL, SIMULATION, REPLAY
ENTITY_STATES	DEVELOPMENT,CM_CONTROLLED
EQUAL_OPERATOR	=, /=
EQUIPMENT_TYPES	UNDEFINED, MILBUS
MIL_BUS_SLOT_CLASS	UNDEFINED, BITS_32, BITS_16, BITS_8, BITS_1, PACKET, NON_STANDARD
PACKET_PART	HEADER,SECONDARY_HEADER,DATA,CHECKSUM

PARAMETER_MODE	IN, OUT, IN_OUT
PROGRAMMING_LANGUAGES	UNDEFINED, ADA, C, ASSEMBLER, OTHER,MIXED
SW_CRITICALITIES	CAT_A, CAT_B, CAT_C, CAT_D
SW_ENTITY_TYPES	UNDEFINED, SWRU, SWEU
SW_TYPE	BOOLEAN_TYPE, STRING_TYPE INTEGER_TYPE, UNSIGNED_INTEGER_TYPE, STATE_CODE_TYPE, BITSET_TYPE, CHARACTER_TYPE, WORD_TYPE, PATHNAME_TYPE, TIME_TYPE, COMPLETION_CODE_TYPE, SUBITEM_PATHNAME_TYPE, PULSE_TYPE, BURST_PULSE_TYPE, REAL_TYPE, LONG_REAL_TYPE
SWRU_TYPE	UNDEFINED, OFFLINE, ONLINE, EMBEDDED
WDU_SYMBOL_TYPE	STATIC, DYNAMIC
WDU_ACTION_TYPE	MENU, HLCL_COMMAND, SYNOPTIC_DISPLAY_REPLACEMENT

Table 7–2 : *Basic MDB data types and their value ranges*

7.6.2.2 Domain and end item association

To each CDU Version a CDU domain is attached specifying the valid end item classes of which end items may be defined within the CDU Version. The CDU domain can be different in several Versions of a CDU.

The domain concept is a mechanism which allows a type based partition of MDB data. The domain assignment serves as a constraint restricting the different types of data available in a CDU.

In the following sections all domains available in CGS together with the related end items are listed.

7.6.2.2.1 Domain CGS

Following end items are available in the domain CGS:

CGS APID
BOOLEAN_MEASUREMENT
BOOLEAN_STIMULUS
BOOLEAN_SW_VARIABLE
BURST_PULSE_STIMULUS
CCSDS_ADU_DESCRIPTION
CCSDS_END_POINT
CPL_SCRIPT
DEMO_ANALOG_STIMULUS
DEMO_BOOLEAN_MEASUREMENT
DEMO_BOOLEAN_SW_VARIABLE
DEMO_BYTE_STREAM_MEASUREMENT
DEMO_DISCRETE_MEASUREMENT
DEMO_DISCRETE_STIMULUS
DEMO_DISCRETE_SW_VARIABLE
DEMO_D_FLOAT_MEASUREMENT
DEMO_D_FLOAT_SW_VARIABLE
DEMO_FLOAT_MEASUREMENT
DEMO_FLOAT_SW_VARIABLE
DEMO_INTEGER_MEASUREMENT
DEMO_INTEGER_SW_VARIABLE
DEMO_TM_PACKET
DEMO_USER_DEFINED_1
DEMO_USER_DEFINED_2
DEMO_USER_DEFINED_3
DEMO_U_INT_MEASUREMENT
DEMO_U_INT_SW_VARIABLE
DOUBLE_FLOAT_MEASUREMENT
DOUBLE_FLOAT_STIMULUS
DOUBLE_FLOAT_SW_VARIABLE
EGSE_ANALOG_STIMULUS
EGSE_BINARY_PACKET
EGSE_STRING_DERIVED_VALUE
EGSE_BYTE_STREAM_MEASUREMENT
EGSE_BYTE_STREAM_SW_VARIABLE

EGSE_DISCRETE_DERIVED_VALUE
EGSE_DISCRETE_MEASUREMENT
EGSE_DISCRETE_STIMULUS
EGSE_DISCRETE_SW_VARIABLE
EGSE_FLOAT_DERIVED_VALUE
EGSE_FLOAT_MEASUREMENT
EGSE_FLOAT_SW_VARIABLE
EGSE_INTEGER_DERIVED_VALUE
EGSE_INTEGER_MEASUREMENT
EGSE_INTEGER_SW_VARIABLE
EGSE_MONITOR_LIST
EGSE_NODE
EGSE_PREDEFINED_TC
EGSE_SOFTWARE
EGSE_TEST_CONFIGURATION
EGSE_USER_MESSAGE
FWDU_COMPOSITE_BINARY
FWDU_CONVERSION_TEXT
FWDU_DATA_DEF_RECORD_TEXT
FWDU_DYNAMIC_OBJECT_TABLE_TEXT
FWDU_EQUIPMENT_CONSTRAINTS
FWDU_GIF_BINARY
FWDU_HELP_TEXT
FWDU_LIBRARY_BINARY
FWDU_LIST_TEXT
FWDU_SYMBOL_BITMAP_BINARY
FWDU_SYMBOL_TABLE_TEXT
FWDU_SYNOPTIC_DISPLAY
FWDU_XWD_BINARY
FWDU_X_BITMAP_BINARY
GDU_DESCRIPTION_LIST
GENERAL_PURPOSES
HLCL_COMMAND_SEQUENCE
INTEGER_CONSTANT
INTEGER_STIMULUS
MX_MODEL
PULSE_STIMULUS
REAL_CONSTANT
REFERENCE_FB
RESPONSE_PACKET
SIMULATED_ADU_DESCRIPTION
STRING_CONSTANT
STRUCTURED_ADU_DESCRIPTION
SWEU
SWOP_COMMAND
SWRU
UCL_AUTOMATED_PROCEDURE
UCL_SYSTEM_LIBRARY

UCL_USER_LIBRARY
UNSIGNED_INTEGER_MEASUREMENT
UNSIGNED_INTEGER_STIMULUS
UNSIGNED_INTEGER_SW_VARIABLE
UNSTRUCTURED_ADU_DESCRIPTION
VIRTUAL
WDU_GROUND_SYMBOL
WDU_GROUND_SYNOPTIC_DISPLAY

Following end items are available in the domain CSS:

Note that the end item type TOPLEVEL_COMPOSITE_FB *can be created by the user, all other end items are created automatically during model saving.*

– TOPLEVEL_COMPOSITE_FB

Following end items are available in the domain CSS_ONBOARD:

BOOLEAN_MEASUREMENT
BOOLEAN_STIMULUS
BOOLEAN_SW_VARIABLE
BURST_PULSE_STIMULUS
DEMO_ANALOG_STIMULUS
DEMO_BOOLEAN_MEASUREMENT
DEMO_BOOLEAN_SW_VARIABLE
DEMO_BYTE_STREAM_MEASUREMENT
DEMO_DISCRETE_MEASUREMENT
DEMO_DISCRETE_STIMULUS
DEMO_DISCRETE_SW_VARIABLE
DEMO_D_FLOAT_MEASUREMENT
DEMO_D_FLOAT_SW_VARIABLE
DEMO_FLOAT_MEASUREMENT
DEMO_FLOAT_SW_VARIABLE
DEMO_INTEGER_MEASUREMENT
DEMO_INTEGER_SW_VARIABLE
DEMO_TM_PACKET
DEMO_U_INT_MEASUREMENT
DEMO_U_INT_SW_VARIABLE
DOUBLE_FLOAT_MEASUREMENT
DOUBLE_FLOAT_STIMULUS
DOUBLE_FLOAT_SW_VARIABLE
INTEGER_STIMULUS
PULSE_STIMULUS
UNSIGNED_INTEGER_MEASUREMENT
UNSIGNED_INTEGER_STIMULUS
UNSIGNED_INTEGER_SW_VARIABLE
VIRTUAL

Following end items are available in the domain EGSE:

CCSDS_ADU_DESCRIPTION
EGSE_ANALOG_STIMULUS
EGSE_BINARY_PACKET
EGSE_STRING_DERIVED_VALUE
EGSE_BYTE_STREAM_MEASUREMENT
EGSE_BYTE_STREAM_SW_VARIABLE
EGSE_DISCRETE_MEASUREMENT
EGSE_DISCRETE_STIMULUS
EGSE_DISCRETE_DERIVED_VALUE
EGSE_DISCRETE_MEASUREMENT
EGSE_DISCRETE_STIMULUS
EGSE_DISCRETE_SW_VARIABLE
EGSE_FLOAT_DERIVED_VALUE
EGSE_FLOAT_MEASUREMENT
EGSE_FLOAT_SW_VARIABLE
EGSE_INTEGER_DERIVED_VALUE
EGSE_MONITOR_LIST
EGSE_NODE
EGSE_PREDEFINED_TC
EGSE_SOFTWARE
EGSE_TEST_CONFIGURATION
EGSE_USER_MESSAGE
GDU_DESCRIPTION_LIST
HLCL_COMMAND_SEQUENCE
RESPONSE_PACKET
SIMULATED_ADU_DESCRIPTION
STRUCTURED_ADU_DESCRIPTION
SWEU
SWOP_COMMAND
UCL_AUTOMATED_PROCEDURE
UCL_USER_LIBRARY
UNSTRUCTURED_ADU_DESCRIPTION
VIRTUAL
WDU_GROUND_SYNOPTIC_DISPLAY

Following end items are available in the domain SDDF:

APID
CCSDS_END_POINT
CPL_SCRIPT
FWDU_COMPOSITE_BINARY
FWDU_CONVERSION_TEXT
FWDU_DATA_DEF_RECORD_TEXT
FWDU_DYNAMIC_OBJECT_TABLE_TEXT
FWDU_EQUIPMENT_CONSTRAINTS
FWDU_GIF_BINARY
FWDU_HELP_TEXT
FWDU_LIBRARY_BINARY
FWDU_LIST_TEXT
FWDU_SYMBOL_BITMAP_BINARY
FWDU_SYMBOL_TABLE_TEXT
FWDU_SYNOPTIC_DISPLAY
FWDU_XWD_BINARY
FWDU_X_BITMAP_BINARY
INTEGER_CONSTANT
MX_MODEL
REAL_CONSTANT
RESPONSE_PACKET
STRING_CONSTANT
SWEU
SWOP_COMMAND
SWRU
VIRTUAL

Following end items are available in the domain UCL_LIBRARY:

INTEGER_CONSTANT
REAL_CONSTANT
STRING_CONSTANT
UCL_AUTOMATED_PROCEDURE
UCL_SYSTEM_LIBRARY
UCL_USER_LIBRARY
VIRTUAL

7.6.3 End Items Description

This section describes the CGS (checkout and test system: VICOS, and CSS) enditems to be stored and maintained in the MDB.

With the enditem description a list of properties(aggregates) is given. Some of the properties are mandatory for checkout and test system purposes, others are not. Optional properties may be present or not.

Some of the enditems have consistency rules globally defined.

Enditems may be used for different purposes. E.g. measurements may be used in the Checkout environment (VICOS/Test Nodes), or in the simulation environment (CSS/ CMAS/Simulation Models). Therefore some aggregates (fields) of the enditems are applicable to one environment only. This is marked within the enditem descriptions.

Note: Under ”Item Properties”, default or standard values are listed where applicable. Defaults marked with (*) mean values, which are taken by CGS (either in the DB or in the load/online software), if no value is given. Standard values not marked are recommended values or values which should be applied (by the user) in most of the cases.

7.6.3.1 Measurements

In the checkout and test system environment, the measurement enditems describe values acquired from the unit under test, front–end equipment or a SAS via the generic input data packet ADU (see below for a description of the ADU mechanism) from a SAS. This includes telemetry data received via CCSDS packets or other telemetry formats.

Several types of measurements and corresponding MDB enditems are available, being

– EGSE_INTEGER_MEASUREMENT
– EGSE_FLOAT_MEASUREMENT
– EGSE_DISCRETE_MEASUREMENT
– EGSE_BYTESTREAM_MEASUREMENT
– UNSIGNED_INTEGER_MEASUREMENT
– DOUBLE_FLOAT_MEASUREMENT
– BOOLEAN_MEASUREMENT
etc

The different types of measurements are defined according to the data type of the engineering value associated with this measurement, i.e. a pressure described as a database enditem will have a floating point value associated with it (e.g. 5.6 bar) whereas a counter will have an integer value. Discrete measurements have state codes as engineering values and data in string format (i.e. ASCII strings) are defined in bytestream measurements.

Enditems declared as measurements can be referred to in UCL APs, procedures or functions, HLCL sequences, synoptic displays, etc. via their pathname.

The overall properties of all measurement enditems in MDB are the same and comprises:

- a *raw value description* which is **mandatory**. This description describes the type of the raw value of a measurement, e.g. data type, size, as acquired in the ADU.
- an *engineering value description* which is **mandatory** for the integer and float measurements but not required for the discrete and bytestream ones. This description describes the type of the engineering value of a measurement, e.g. value ranges.
- a *calibration description* which is **mandatory**. This description describes the transformation of the described raw value into the (described) engineering value during run–time, e.g. a calibration of an integer value via a polynom into a floating point value.
- an *engineering value logging definition* which is **optiona**l. This definition allows to define in MDB whether the value shall be logged whenever being updated later during run–time.
- a *physical address*, which is **optional**. This information is needed by the SAS to physically acquire the value and is not processed by CGS. This information is optional (for an individual measurement) since a SAS may know how to acquire the data or may use global physical address information stored with the ADU description.
- a *monitoring definition* which is **optional**. This property defines the limits against which the actual value of the measurement shall be checked and which actions shall be taken in case the limits are violated (out of limit situation)

7.6.3.1.1 EGSE_INTEGER_MEASUREMENT

This enditem type describes a measurement acquired via the ADU service from a SAS with an *integer* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
RAW VALUE TYPE	SIGNED_INTEGER	MANDATORY FLOAT and BYTESTREAM not supported by VICOS UNSIGNED_INTEGER: Raw_Value_Size: 1..31
RAW VALUE SIZE IN BITS	32	mandatory Allowed Range: 1..32 Dependent on Raw_Value_Type range for UNSIGNED_INTEGER: 1..31
INTEGER RAW VALUE RANGE	MIN_INTEGER .. MAX_INTEGER	mandatory (if raw value type integer) Dependent on size MIN_INTEGER <= A < B <= 2**size–1
UNSIGNED INTEGER RAW VALUE RANGE	0 .. MAX_UNSIGN	mandatory (if raw value type unsigned) Dependent on size: 0 <= A < B <= 2**size–1
engineering value description		
INTEGER ENGINEERING RANGE	MIN_INTEGER .. MAX_INTEGER(*)	Dependent on Raw Value Range and Calibration: INTEGER_ENGINEERING_RANGE = Calibration(RAW_VALUE_RANGE)
EVL CONTROL	FALSE(*)	Activates logging of measurement in online checkout.Initially, engineering value logging should be disabled

Aggregates	Standard Value / Default (*)	Comment
analog calibration		
CURVE TYPE	POLYNOM	mandatory POINT_PAIRS,POLYNOM,IDENT-ICAL For VICOS: IDENTICAL means same as POLYNOM with coefficients 0.0,1.0,0.0,0.0,0.0
ANALOG POINT PAIRS	(MIN_INTEGER, MIN_INTEGER) (MAX_INTEGER, MAX_INTEGER) (*)	mandatory (if curve type = point pairs) The 2nd value must be within the limits defined by the engineering value range. The sequence of 2nd values must be monotonic over the specified eng. value range i.e. ENG_VALUEi <= ENG_VALUEi+1 or ENG_VALUEi >= ENG_VALUEi+1 for i=1 .. N–1
ANALOG CALIBRATION COEF-FICIENTS	0.0, 1.0, 0.0, 0.0, 0.0 (*)	mandatory (if curve type = polynom)
ANALOG DECAL POINT PAIRS	(MIN_INTEGER, MIN_INTEGER) (MAX_INTEGER, MAX_INTEGER) (*)	optional used by CSS (Simulation) only
Monitoring Description		Defines limits and actions for nominal monitoring and danger monitoring. Note: If danger limits are defined, nominal limits must be defined too.
INTEGER NOMINAL LIMITS All 5 limit sets Nominal_Low_Action Nominal_High_Action Nominal_Delta_Action Nominal_Low_Message Nominal_High_Message Nominal_Delta_Message	 <empty> (*) <empty> (*) <empty> (*)	optional If any limit is defined, high and low limits must be defined Delta limits are optional. If any, limit set 1 must be defined. AP or STIMULUS/PREDEFINED_TC/ BINARY_PACKET maybe specified by Pathname Additional message text which is dis-played in the message window when an limit violation occurs, may be specified by pathname

Aggregates	Standard Value / Default (*)	Comment
INTEGER DANGER LIMITS High and Low Limit, Delta Limit Danger_Low_Action Danger_High_Action Danger_Delta_Action Danger_Low_Message Danger_High_Message Danger_Delta_Message	 <empty> (*) <empty> (*) <empty> (*)	optional High and low limits must be defined Delta limits are optional. AP or STIMULUS/PREDEFINED_TC/ BINARY_PACKET maybe specified by Pathname Additional message text which is displayed in the message window when an limit violation occurs, specified by pathname
MAX ALARM COUNTER	1 (*)	MAX ALARM COUNTER = n means that after n times of consecutive limit violations the exception for nominal limits will be raised
Conditions		

Aggregates	Standard Value / Default (*)	Comment
INTEGER_CONDITIONS	empty	optional Describes conditions, which define/trigger an action when they are becoming true. A Condition is defined by expression: "{this enditem} <Operator> <Value_1> [.. Value_2]" e.g.: "{this enditem} > 10" or: 10 < {this enditem} < 100
Operator	empty	in_range, =, /=,>,< ,<=,>=
Value_1_(Low_Value)	empty	Expected Value. For in_range: Low value of range
Value_2_(High_Value)	empty	If in_range: High value of range
Action_Type	empty	ENABLE_PROCESSING, SWITCH_LIMIT_SET, START_AP
Action Enditem Reference	empty	The item to be enabled, the AP to be started resp. the enditem which shall have the limit set switched. Enditem Type: if Action_Type = START_AP UCL_AUTOMATED_PROCEDURE Else: EGSE_..._MEASUREMENT EGSE_..._SW_VARIABLE EGSE_..._DERIVED_VALUE, VIRTUAL, CDU, ADU, MONITOR_LIST
Limit Set Number	0	0..5, if Action =SWITCH_LIMIT_SET Number of Limit set to be selected for Action Enditem Reference

Aggregates	Standard Value / Default (*)	Comment
EGSE PHYSICAL ADDRESS		optional Describes the physical acquisition address of the raw value
Front–end Node Address	empty string	Address of the front–end equipment, e.g. an internet address in case the front–end is composed of an intelligent unit controlling several non–intelligent devices. This attribute is not used by CMAS.
Front–end Bus	0	Identification of the bus connecting the frontend equipment or the unit under test, e.g. a Mil–Bus number. For CMAS, this attribute identifies the MILBus number and is mandatory.
Device Type	empty string	Type of the acquisition device, e.g. HP3852A. For CMAS, this attribute is mandatory and must have the value ”MIL_BUS”. (Note: value is stored as STRING in the MDB; default is ””, i.e empty string)
Device Address	empty string	Address of the acquisition device, e.g. inside the front end equipment. Can be an IEEE488 address or a remote terminal address of a MilBus, for example. For CMAS, this attribute is mandatory and denotes the MIL–Bus remote terminal address, i.e. its value must be an unsigned integer in the range 0..31. Addresses 0 and 31 are reserved for special purposes and should not be used. For COF EGSE, this field denotes the logical name of the MIL Bus
Device Subaddress	0	Subaddress of the acquisition device, e.g. an IEEE488 subaddress or a remote terminal subaddress on the MilBus. For CMAS, this attribute is mandatory and denotes the MIL–Bus remote terminal subaddress, i.e. its value must be an unsigned integer in the range 0..31.

Consistency Checks:

Nominal High Limit < Danger High Limit <= High_Value in the engineering description

Nominal Low Limit > Danger Low Limit => Low Value of the engineering description

The referenced AP action in monitoring limits or conditions must not have parameters.

If the referenced stimulus/TC/BP action in monitoring limits has parameters, default values must be assigned to each. Default values will be used when they are sent.

Low Limit < High Limit

If Action_Type = ’Switch_Limit_Set’, the referenced enditem must have nominal limit sets defined

If Action_Type = ’Switch_Limit_Set’, the given Limit Set Number must be defined for the Action_Enditem_Reference

If Operator = In_Range: Value_2 > Value_1

The Action_Enditem_Reference must be within the allocated set of enditems for the test node, where the enditem carrying the condition is allocated to.

Notes:

(1) An enditem is enabled for processing, if at least one condition within the whole test node is true, that defines processing for this enditem.

(2) An AP is started once when the condition becomes true. On subsequent new samples of the enditem leading to true value for the condition, no furhter AP start is performed. After being false and true again, the AP is started again.

7.6.3.1.2 EGSE_FLOAT_MEASUREMENT

This enditem type describes a measurement acquired via the ADU service from a SAS with a *float* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
RAW VALUE TYPE	FLOAT	MANDATORY BYTESTREAM not supported by VICOS
RAW VALUE SIZE IN BITS	32	mandatory Allowed Range: Raw Value Type FLOAT: 32 Raw Value Type INTEGER: 1..32 RVType UNSIGNED_INTEGER: 1..31
FLOAT RAW VALUE RANGE	MIN_FLOAT .. MAX_FLOAT (*)	optional (if raw value type = float) Dependent on size
INTEGER RAW VALUE RANGE	MIN_INTEGER .. MAX_INTEGER (*)	optional (if raw value type integer) Dependent on size
UNSIGNED INTEGER RAW VALUE RANGE	0 .. MAX_UNSIGNED(*)	optional (if raw value type unsigned) Dependent on size
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical acquisition address of the raw value
engineering value description		
ENGINEERING UNITS		optional
FLOAT ENGINEERING RANGE	MIN_FLOAT .. MAX_FLOAT (*)	Dependent on Raw Value Range and Calibration: FLOAT_ENGINEERING_RANGE = Calibration(RAW_VALUE_RANGE)
EVL CONTROL	FALSE	Activates logging of measurement in online checkout.Initially, engineering value logging should be disabled

Aggregates	Standard Value / Default (*)	Comment
analog calibration		
CURVE TYPE	POLYNOM	mandatory POINT_PAIRS,POLYNOM, IDENT-ICAL For VICOS: IDENTICAL means same as POLYNOM with coefficients 0.0,1.0,0.0,0.0,0.0
ANALOG POINT PAIRS	(MIN_FLOAT, MIN_FLOAT) (MAX_FLOAT, MAX_FLOAT)	mandatory (if curve type = point pairs) The 2nd value must be within the limits defined by the engineering value range. The sequence of 2nd values must be monotonic over the specified eng. value range i.e. ENG_VALUEi <= ENG_VALUEi+1 or ENG_VALUEi >= ENG_VALUEi+1 for i=1 .. N–1
ANALOG CALIBRATION COEF-FICIENTS	0.0, 1.0, 0.0, 0.0, 0.0	mandatory (if curve type = polynom)
ANALOG DECAL POINT PAIRS	(MIN_FLOAT, MIN_FLOAT) (MAX_FLOAT, MAX_FLOAT	optional used by CSS (Simulation) only
Monitoring Description		Defines limits and actions for nominal monitoring and danger monitoring. Note: If danger limits are defined, nominal limits must be defined too.
FLOAT NOMINAL LIMITS All 5 limit sets Nominal_Low_Action Nominal_High_Action Nominal_Delta_Action Nominal_Low_Message Nominal_High_Message Nominal_Delta_Message	 <empty> <empty> <empty>	optional If any limit is defined, high and low limits must be defined Delta limits are optional. If any, limit set 1 must be defined. AP or STIMULUS/PREDEFINED_TC/ BINARY_PACKET maybe specified by Pathname Additional message text which is dis-played in the message window when an limit violation occurs, may be specified by pathname

Aggregates	Standard Value / Default (*)	Comment
FLOAT DANGER LIMITS High Limit, Low Limit, Delta LLimit Danger_Low_Action Danger_High_Action Danger_Delta_Action Danger_Low_Message Danger_High_Message Danger_Delta_Message	 <empty> <empty> <empty>	optional If any limit is defined, high and low limits must be defined Delta limits are optional. AP or STIMULUS/PREDEFINED_TC/ BINARY_PACKET maybe specified by Pathname Additional message text which is displayed in the message window when an limit violation occurs, may be specified by pathname
MAX ALARM COUNTER	1 (*)	MAX ALARM COUNTER = n means that after n times of consecutive limit violations the exception for nominal limits will be raised
Conditions		
FLOAT_CONDITIONS Operator Value_1_(Low_Value) Value_2_(High_Value) Action_Type Action Enditem Reference Limit Set Number	empty	optional See INTEGER_CONDITION in EGSE_INTEGER_MEASURMENT

Consistency Checks:

Danger High Limit <= High Value in the engineering description

Danger Low Limit => Low Value of the engineering description

The referenced AP action in monitoring limits or condition must not have parameters.

If the referenced stimulus/TC/BP action in monitoring limits has parameters, default values must be assigned to each. Default values will be used when they are sent.

Low Limit < High Limit

If Action_Type = 'Switch_Limit_Set', the referenced enditem must have nominal limit sets defined

If Action_Type = 'Switch_Limit_Set', the given Limit Set Number must be defined for the Action_Enditem_Reference

If Operator = In_Range: Value_2 > Value_1

The Action_Enditem_Reference must be within the allocated set of enditems for the test node, where the enditem carrying the condition is allocated to.

Notes: For notes onconditions refer to EGSE_INTEGER_MEASUREMENT

7.6.3.1.3 EGSE_DISCRETE_MEASUREMENT

This enditem type describes a measurement acquired,e.g. via ADU service from a SAS with a *discrete* engineering value. This end item is also the counterpart to the CSS top level I/O of type state code.
The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

This End Item is also used for acquisition of boolean values of a 1-Bit digital measurement device from SAS interfacing to frontends or measurement devices, or sent within telemetry packets (i.e. all values acquired via SAS).

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
RAW VALUE TYPE	UNSIGNED_INTEGER	MANDATORY UNSIGNED_INTEGER: Others are not supported !!!
RAW VALUE SIZE IN BITS	31	mandatory Allowed Range: 1..31
UNSIGNED INTEGER RAW VALUE RANGE	0 .. MAX_UNSIGN (*)	optional Dependent on size
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical acquisition address of the raw value
engineering value description		
EVL CONTROL	FALSE(*)	Activates logging of measurement in online checkout.Initially, engineering value logging should be disabled
discrete calibration		
DISCRETE_CALIBRATION up to 256 Statecode / Raw Value Specifications. A Raw Value Spec. is a single value or a range: Raw_Value..High_Value_of_Range	if empty, the OTHERS statecode applies for any raw value	optional Statecodes are strings up to 8 characters, starting with a letter and containing letters/digits and underscores
UNDEFINED_VALUES_STATE_CODE	OTHERS (*)	optional Defines statecode for all raw values which do not appear in the list of 256 statecodes above

Aggregates	Standard Value / Default (*)	Comment
Monitoring Description		
DISCRETE MONITOR LIST up to 5 Monitoring Sets: Expected Value Exception Message Exception Action		 Expected Statecode one of the state codes defined in the DISCRETE CALIBRATION Message to be generated if expected value is not the same as the actual value GDU/AP to be sent/started if expected value is not the same as the actual value
MAX ALARM COUNTER	1 (*)	MAX ALARM COUNTER = n means that after n times of consecutive expected value violations the exception for nominal limits will be raised
Conditions		
DISCRETE_CONDITIONS Operator Value Action_Type Action Enditem Reference Limit Set Number	empty	optional See INTEGER_CONDITIONS in EGSE_INTEGER_MEASURMENT E.g.”/=” ”Expected String” means, the condition is true, if the value of this enditem does not have the value Expected String. =, /= Expected String. See INTEGER_CONDITION in EGSE_INTEGER_MEASURMENT

Consistency Checks:

The referenced AP action in monitoring limits or conditions must not have parameters.

If the referenced stimulus/TC/BP action in monitoring limits has parameters, default values must be assigned to each. Default values will be used when they are sent.

The expected value must be defined as statecode in the discrete calibration.

If discrete conditions are specified, the expected string must be defined in the discrete calibration

If Action_Type = 'Switch_Limit_Set', the referenced enditem must have nominal limit sets defined

If Action_Type = 'Switch_Limit_Set', the given Limit Set Number must be defined for the Action_Enditem_Reference

The Action_Enditem_Reference must be within the allocated set of enditems for the test node, where the enditem carrying the condition is allocated to.

The relation (Low_)Raw_Value < High_Raw_Value must be fulfilled for each record, if High_Raw_Value is given.

The defined raw value ranges must not overlap

Notes: For notes on conditions refer to EGSE_INTEGER_MEASUREMENT

7.6.3.1.4 EGSE_BYTESTREAM_MEASUREMENT

This enditem type describes a measurement acquired via the ADU service from an SAS with a *bytestream* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
RAW VALUE TYPE	UNSIGNED_INTEGER	MANDATORY BYTE_STREAM: Others are not supported !!!
RAW VALUE SIZE IN BYTES	255	mandatory Allowed Range: 1..255
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)	all fields empty (*)	optional Describes the physical acquisition address of the raw value
engineering value description		
EVL CONTROL	FALSE(*)	Activates logging of measurement in online checkout.Initially, engineering value logging should be disabled
calibration		
BYTESTREAM_CALIBRATION POSITION LENGTH	 1 255	optional defines position within string (in bytes) Range: 1..255 defines length of substring to be extracted (in bytes) Range: 1..255
Monitoring Description		
BYTESTREAM MONITOR LIST up to 5 Monitoring Sets: Expected Value Exception Message Exception Action		 Expected String (length as defined in BYTESTREAM CALIBRATION) Message to be generated if expected value is not the same as the actual value GDU/AP to be sent/started if expected value is not the same as the actual value
MAX ALARM COUNTER	1 (*)	MAX ALARM COUNTER = n means that after n times of consecutive expected value violations the exception for nominal limits will be raised

Aggregates	Standard Value / Default (*)	Comment
Conditions		
STRING_CONDITIONS Operator Value Action_Type Action Enditem Reference Limit Set Number	empty	optional See DISCRETE_CONDITION in EGSE_DISCRETE_MEASUR- MENT

Consistency Checks:

The referenced AP action in monitoring limits or conditions must not have parameters.

If the referenced stimulus/TC/BP action in monitoring limits has parameters, default values must be assigned to each. Default values will be used when they are sent.

If Action = 'Switch_Limit_Set', the referenced enditem must have nominal limit sets defined

If Action = 'Switch_Limit_Set', the given Limit Set Number must be defined for the Action_Enditem_Reference

The Action_Enditem_Reference must be within the allocated set of enditems for the test node, where the enditem carrying the condition is allocated to.

Notes: For notes on conditions refer to EGSE_INTEGER_MEASUREMENT

7.6.3.1.5 DOUBLE_FLOAT_MEASUREMENT

This enditem type describes a measurement acquired with a *double float* engineering value. It is always calibrated from integer or unsigned integer values.

The enditem type is only used for CSS Models.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
RAW VALUE TYPE	Integer	Integer or Unsigned_Integer
RAW VALUE SIZE IN BITS	32	raw value type integer: 1..32 raw value type unsigned_integer: 1..31
INTEGER RAW VALUE RANGE	MIN_INTEGER .. MAX_INTEGER (*)	mandatory (if raw value type integer) Dependent on size
UNSIGNED INTEGER RAW VALUE RANGE	0 .. MAX_UNSIGN (*)	mandatory (if raw value type unsigned) Dependent on size
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)	all fields empty (*)	optional Describes the physical acquisition address of the raw value
engineering value description		
ENGINEERING UNITS		optional
DOUBLE_FLOAT ENG RANGE	MIN_DOUBLE_FLOAT .. MAX_DOUBLE_FLOAT (*)	Dependent on Raw Value Range and Calibration: FLOAT_ENGINEERING_RANGE = Calibration(RAW_VALUE_RANGE)
analog calibration		
CURVE TYPE	POINT_PAIRS	mandatory
ANALOG POINT PAIRS	(MIN_DOUBLE_FLOAT, MIN_DOUBLE_FLOAT) (MAX_DOUBLE_FLOAT, MAX_DOUBLE_FLOAT)	mandatory (if curve type = point pairs)
ANALOG CALIBRATION COEFFICIENTS	0.0, 1.0, 0.0, 0.0, 0.0	mandatory (if curve type = polynom)
ANALOG DECAL POINT PAIRS		optional (not used)
Monitoring Description		
DOUBLE_FLOAT NOMINAL LIMITS		optional (not used)
DOUIBLE_FLOAT DANGER LIMITS		optional (not used)
MAX ALARM COUNTER		optional (not used)

7.6.3.1.6 UNSIGNED_INTEGER_MEASUREMENT

This enditem type describes a measurement acquired with an *unsigned_integer* engineering value. The enditem type is only used for CSS Models.

Aggregates	Standard Value / Default (*)	Comment
raw value description		
RAW VALUE TYPE	UNSIGNED_INTEGER	Only unsigned allowed
RAW VALUE SIZE IN BITS	31	mandatory Allowed Range: 1..31
UNSIGNED INT RAW VALUE RANGE	0 .. MAX_UNSIGN	mandatory (if raw value type unsigned) Dependent on size
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical acquisition address of the raw value
engineering value description		
ENGINEERING UNITS		optional
UNSIGNED_INTEGER_ENGINEERING_RANGE	0.. MAX_UNSIGNED	Dependent on Calibration: UNSIGNED_INTEGER_ENGINEERING_RANGE= Calibration(UNSIGNED_INTEGER_RAW_RANGE)
analog calibration		
CURVE TYPE	POINT_PAIRS	mandatory
ANALOG POINT PAIRS	(0, 0) (MAX_UNSIGNED, MAX_UNSIGNED)	mandatory (if curve type = point pairs)
ANALOG CALIBRATION COEFFICIENTS	0.0, 1.0, 0.0, 0.0, 0.0	mandatory (if curve type = polynom)
ANALOG DECAL POINT PAIRS		optional (not used)
Monitoring Description		
UNSIGNED_INT NOMINAL LIMITS		optional (not used)
UNSIGNED_INT DANGER LIMITS		optional (not used)
MAX ALARM COUNTER		optional (not used)

7.6.3.1.7 BOOLEAN_MEASUREMENT

This enditem type describes a measurement acquired with a *boolean* engineering value.
The enditem type is only used for CSS Models.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
RAW VALUE TYPE	INTEGER	MANDATORY INTEGER:
RAW VALUE SIZE IN BITS	1	mandatory Allowed Range: 1
INTEGER RAW VALUE RANGE	MIN_INTEGER .. MAX_INTEGER	mandatory Dependent on size
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical acquisition address of the raw value
EVL CONTROL		optional (not used)
Boolean calibration		
BOOLEAN CALIBRATION		mandatory
Monitoring Description		
BOOLEAN_MONITORING		optional (not used)

7.6.3.2 SW Variables

In the checkout environment, the SW variable enditems describe values locally maintained in the Test Nodes. This includes

– internal status data (maintained in Ada) of the products TES, the SW running on test nodes, and DBS (the central test result database)
– user defined SW variables which are then available in UCL/HLCL as place holders for writing/reading data. SW variables can be read (any type) or written (only the UCL SW variables not mapped to internal status data) from SAS also: Software Variables may be updated via the TES_API using the WRITE_VALUE services (i.e. by directly writing values to end items).

Enditems declared as SW variables can be referred to in UCL APs, procedures or functions, HLCL sequences, synoptic displays, SAS etc. via their pathname.

Several sub–types of SW variables are available, being

– EGSE_INTEGER_SW_VARIABLE
– EGSE_FLOAT_SW_VARIABLE
– EGSE_DISCRETE_SW_VARIABLE
– EGSE_BYTESTREAM_SW_VARIABLE

The different types of SW variables are defined according to the data type of the engineering value associated with it, i.e. a pressure described as a database enditem will have a floating point value associated with it (e.g. 5.6 bar) whereas a counter will have an integer value. Discrete SW variables have state codes as engineering values and data in string format (i.e. ASCII strings) are defined in bytestream SW variables.

The overall properties of all SW variable enditems in MDB is the same and comprises:

- an *initial value description* which is **mandatory**. This description defines the initial value of the SW variable. This is necessary because a value can be assigned from UCL/HLCL and until this happens for the first time the value would be undefined.
- an *engineering value description* which is **mandatory** for the integer and float SW variables but not required for the discrete and bytestream ones. This description describes the type of the engineering value of a measurement, e.g. value ranges.
- an *HK source definition* which is **optiona**l. This property defines to which housekeeping value the SW variable shall be bound to. Housekeeping values are internal status data of Ada SW (CGS Processes) A list of all available housekeeping values which can be linked with SW variables can be found in teh resp. chapter of this document.
- an *engineering value logging definition* which is **optional**. This definition allows to define in MDB whether the value shall be logged whenever being updated later during run–time. (Note: This flag can be overwritten by online UCL commands)
- a *monitoring definition* which is **optional**. This property defines the limits against which the actual value of the SW variable shall be checked and which actions shall be taken in case the limits are violated (out of limit situation)

7.6.3.2.1 EGSE_INTEGER_SW_VARIABLE

This enditem type describes a SW variable with an *integer* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
INITIAL INTEGER VALUE		The value initially defined for the SW Variable.
HK SOURCE	empty (*)	optional Integer number for HK ID if value of SW variable is to be fetched from it.. The SW variable is writable by UCL/ HLCL or SAS only, if HK_SOURCE is empty (i.e. no HK ID is assigned) In case an initial value is specified, it will be ignored for SW variables bound to HK Sources.
engineering value description		
INTEGER ENGINEERING RANGE	MIN_INTEGER .. MAX_INTEGER(*)	optional
EVL CONTROL	FALSE(*)	Activates logging of variable in online checkout.
Monitoring Description		Defines limits and actions for nominal monitoring and danger monitoring. Note: If danger limits are defined, nominal limits must be defined too.
INTEGER NOMINAL LIMITS All 5 limit sets Nominal_Low_Action Nominal_High_Action Nominal_Delta_Action Nominal_Low_Message Nominal_High_Message Nominal_Delta_Message	 <empty> (*) <empty> (*) <empty> (*)	optional If any limit is defined, high and low limits must be defined Delta limits are optional. If any, limit set 1 must be defined. AP or STIMULUS/PREDEFINED_TC/ BINARY_PACKET maybe specified by Pathname Additional message text which is displayed in the message window when an limit violation occurs, specified by pathname

Aggregates	Standard Value / Default (*)	Comment
INTEGER DANGER LIMITS Low Limit,High Limit,Delta Limit Danger_Low_Action Danger_High_Action Danger_Delta_Action Danger_Low_Message Danger_High_Message Danger_Delta_Message	 <empty> (*) <empty> (*) <empty> (*)	optional If any limit is defined, high and low limits must be defined Delta limits are optional. AP or STIMULUS/PREDEFINED_TC/ BINARY_PACKET maybe specified by Pathname Additional message text which is displayed in the message window when an limit violation occurs, specified by pathname
MAX ALARM COUNTER	1 (*)	MAX ALARM COUNTER = n means that after n times of consecutive limit violations the exception for nominal limits will be raised
Conditions		
INTEGER_CONDITIONS Operator Value_1_(Low_Value) Value_2_(High_Value) Action_Type Action Enditem Reference Limit Set Number	empty	optional See INTEGER_CONDITION in EGSE_INTEGER_MEASUR-MENT

Consistency Checks:

Danger High Limit <= High Value in the engineering description

Danger Low Limit >= Low Value of the engineering description

The referenced AP action in monitoring limits or conditions must not have parameters.

If the referenced stimulus/TC/BP action in monitoring limits has parameters, default values must be assigned to each. Default values will be used when they are sent.

Low Limit < High Limit

If Action_Type = 'Switch_Limit_Set', the referenced enditem must have nominal limit sets defined

If Action_Type = 'Switch_Limit_Set', the given Limit Set Number must be defined for the Action_Enditem_Reference

If Operator = In_Range: Value_2 > Value_1

The Action_Enditem_Reference must be within the allocated set of enditems for the test node, where the enditem carrying the condition is allocated to.

Notes: For notes onconditions refer to EGSE_INTEGER_MEASUREMENT

7.6.3.2.2 EGSE_FLOAT_SW_VARIABLE

This enditem type describes a SW_VARIABLE with a *float* engineering value.

This End Item is used for acquisition of values from software (e.g. AP, SAS), which engineering unit is a float. This is updated via the TES_API using the WRITE_VALUE services (i.e. by directly writing values to end items), or for values that are generated internally (housekeeping values).

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
INITIAL_FLOAT_VALUE	0.0	The value initially defined for the SW Variable
HK SOURCE	empty (*)	optional Integer number for VICOS HK ID if value of SW variable is to be fetched from it. The SW variable is writable by UCL/ HLCL or SAS only, if HK_SOURCE is empty (i.e. no HK ID is assigned)
engineering value description		
ENGINEERING UNITS		optional
FLOAT ENGINEERING RANGE	MIN_FLOAT .. MAX_FLOAT (*)	optional
EVL CONTROL	FALSE (*)	Activates logging of measurement in online checkout.
Monitoring Description		Defines limits and actions for nominal monitoring and danger monitoring. Note: If danger limits are defined, nominal limits must be defined too.
FLOAT NOMINAL LIMITS All 5 limit sets Nominal_Low_Action Nominal_High_Action Nominal_Delta_Action Nominal_Low_Message Nominal_High_Message Nominal_Delta_Message	 <empty> <empty> <empty>	optional If any limit is defined, high and low limits must be defined Delta limits are optional. If any, limit set 1 must be defined. AP or STIMULUS/PREDEFINED_TC/ BINARY_PACKET maybe specified by Pathname Additional message text which is dis- played in the message window when an limit violation occurs, may be specified by pathname

Aggregates	Standard Value / Default (*)	Comment
FLOAT DANGER LIMITS Low Limit,High Limit,Delta Limit Danger_Low_Action Danger_High_Action Danger_Delta_Action Danger_Low_Message Danger_High_Message Danger_Delta_Message	 <empty> (*) <empty>m (*) <empty> (*)	optional If any limit is defined, high and low limits must be defined Delta limits are optional. AP or STIMULUS/PREDEFINED_TC/ BINARY_PACKET maybe specified by Pathname Additional message text which is displayed in the message window when an limit violation occurs, may be specified by pathname
MAX ALARM COUNTER	1 (*)	MAX ALARM COUNTER = n means that after n times of consecutive limit violations the exception for nominal limits will be raised
Conditions		
FLOAT_CONDITIONS Operator Value_1_(Low_Value) Value_2_(High_Value) Action_Type Action Enditem Reference Limit Set Number	empty	optional See INTEGER_CONDITION in EGSE_INTEGER_MEASURMENT

Consistency Checks:

Danger High Limit <= High Value in the engineering description

Danger Low Limit => Low Value of the engineering description

The referenced AP action in monitoring limits must not have parameters.

If the referenced stimulus/TC/BP action in monitoring limits has parameters, default values must be assigned to each. Default values will be used when they are sent.

Low Limit < High Limit

If Action = ’Switch_Limit_Set’, the referenced enditem must have nominal limit sets defined

If Action = ’Switch_Limit_Set’, the given Limit Set Number must be defined for the Action_Enditem_Reference

If Operator = In_Range: Value_2 > Value_1

The Action_Enditem_Reference must be within the allocated set of enditems for the test node, where the enditem carrying the condition is allocated to.

Notes: For notes onconditions refer to EGSE_INTEGER_MEASUREMENT

7.6.3.2.3 EGSE_DISCRETE_SW_VARIABLE

This enditem type describes a SW_VARIABLE with a *state code* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an AP.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
INITIAL_DISCRETE_VALUE	”” (empty)	The value initially defined for the SW Variable
HK SOURCE	empty (*)	optional Integer number for VICOS HK ID if value of SW variable is to be fetched from it. The SW variable is writable by UCL/ HLCL or SAS only, if HK_SOURCE is empty (i.e. no HK ID is assigned)
Engineering value description		
EVL CONTROL	FALSE(*)	Activates logging of measurement in online checkout.Initially, engineering value logging should be disabled
Discrete calibration		for use by GWDU only !
DISCRETE_CALIBRATION up to 256 Statecode / Raw Value Specifications. A Raw Value Spec. is a single value or a range: Raw_Value..High_Value_of_Range	if empty, the OTHERS statecode applies for any raw value	optional Statecodes are strings up to 8 characters, starting with a letter and containing letters/digits and underscores Raw values are not applicable for Derived Values
UNDEFINED_VALUES_STATE_CODE	OTHERS (*)	optional N/A for SW_variables
Monitoring Description		
DISCRETE MONITOR LIST Up to 5 Limit Sets Expected Value Exception Message Exception Action	empty (*)	one of the state codes defined in the DISCRETE CALIBRATION Expected Statecode/String Message to be generated if expected value is not the same as the actual value GDU/AP to be sent/started if expected value is not the same as the actual value
MAX ALARM COUNTER	1 (*)	MAX ALARM COUNTER = n means that after n times of consecutive expected value violations the exception for nominal limits will be raised

Aggregates	Standard Value / Default (*)	Comment
Conditions		
DISCRETE_CONDITIONS Operator Value Action_Type Action Enditem Reference Limit Set Number	empty	optional See DISCRETE_CONDITION in EGSE_DISCRETE_MEASUR-MENT

Consistency Checks:

The referenced AP action in monitoring limits must not have parameters.

If the referenced stimulus/TC/BP action in monitoring limits has parameters, default values must be assigned to each. Default values will be used when they are sent.

If discrete conditions are specified, the expected string must be defined in the discrete calibration

If Action_Type = ’Switch_Limit_Set’, the referenced enditem must have nominal limit sets defined

If Action_Type = ’Switch_Limit_Set’, the given Limit Set Number must be defined for the Action_Enditem_Reference

The Action_Enditem_Reference must be within the allocated set of enditems for the test node, where the enditem carrying the condition is allocated to.

Notes: For notes on conditions refer to EGSE_INTEGER_MEASUREMENT

7.6.3.2.4 EGSE_BYTESTREAM_SW_VARIABLE

This enditem type describes a SW_VARIABLE with a *bytestream* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
INITIAL_BYTESTREAM_VALUE	”” (empty)	The string value initially defined for the SW Variable Note: Due to a bug, this aggregate is currently defined as a multirecord aggregate. Only the first entry of this agrregate is used, however.
HK SOURCE	empty (*)	optional Integer number for VICOS HK ID if value of SW variable is to be fetched from it. The SW variable is writable by UCL/HLCL or SAS only, if HK_SOURCE is empty (i.e. no HK ID is assigned)
engineering value description		
EVL CONTROL	FALSE(*)	Activates logging of measurement in online checkout.
Monitoring Description		
BYTESTREAM MONITOR LIST Up to 5 Limit Sets: Expected Value Exception Message Exception Action	empty (*)	 Expected String Message to be generated if expected value is not the same as the actual value GDU/AP to be sent/started if expected value is not the same as the actual value
MAX ALARM COUNTER	1 (*)	MAX ALARM COUNTER = n means that after n times of consecutive expected value violations the exception for nominal limits will be raised
Conditions		
STRING_CONDITIONS Operator Value Action_Type Action Enditem Reference Limit Set Number	empty	optional See DISCRETE_CONDITION in EGSE_DISCRETE_MEASURMENT

Consistency Checks:

The referenced AP action in monitoring limits must not have parameters.

If the referenced stimulus/TC/BP action in monitoring limits has parameters, default values must be assigned to each. Default values will be used when they are sent.

If Action = 'Switch_Limit_Set', the referenced enditem must have nominal limit sets defined

If Action = 'Switch_Limit_Set', the given Limit Set Number must be defined for the Action_Enditem_Reference

The Action_Enditem_Reference must be within the allocated set of enditems for the test node, where the enditem carrying the condition is allocated to.

Notes: For notes on conditions refer to EGSE_INTEGER_MEASUREMENT

7.6.3.2.5 UNSIGNED_INTEGER_SW_VARIABLE

This enditem type describes a SW variable with an *unsigned integer* engineering value.
The enditem type is only used for CSS Models.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
INITIAL_UNSIGNED_INTEGER VALUE		The value initially defined for the SW Variable
engineering value description		
UNSIGNED_INTEGER_ENGIN-EERING RANGE	0 .. MAX_UNSIGNED	
Monitoring Description		
UNSIGNED_INT_NOMINAL LI-MITS		optional (not used)
UNSIGNED_INTEGER_DANGER LIMITS		optional (not used)
MAX ALARM COUNTER		optional (not used)

7.6.3.2.6 DOUBLE_FLOAT_SW_VARIABLE

This enditem type describes a SW variable with an *double float* engineering value.
The enditem type is only used for CSS Models.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
INITIAL_DOUBLE_FLOAT VALUE		The value initially defined for the SW Variable
engineering value description		
DOUBLE_FLOAT_ENGINEER-ING RANGE	MIN_DOUBLE_FLOAT .. MAX_DOUBLE_FLOAT	
Monitoring Description		
DOUBLE_FLOAT_NOMINAL LI-MITS	empty (*)	optional (not used)
DOUBLE_FLOAT_IN-TEGER_DANGER LIMITS	empty (*)	optional (not used)
MAX ALARM COUNTER	1 (*)	optional (not used)

7.6.3.2.7 BOOLEAN_SW_VARIABLE

This enditem type describes a SW variable with an *boolean* engineering value.
The enditem type is only used for CSS Models.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
raw value description		
INITIAL_BOOLEAN_VALUE	FALSE	The value initially defined for the SW Variable
engineering value description		
EVL CONTROL		optional (not used)
Monitoring Description		
UNSIGNED_INT_NOMINAL LIMITS		optional (not used)
UNSIGNED_INTEGER_DANGER LIMITS		optional (not used)
MAX ALARM COUNTER		optional (not used)

7.6.3.3 Derived Values

In the checkout environment, Derived Value end items describe values calculated according to a mathematical/logical expression. A derived value can be displayed and monitored as any measurement and software variable.

The following types of Derived Values are available:

– EGSE_INTEGER_DERIVED_VALUE
– EGSE_FLOAT_DERIVED_VALUE
– EGSE_DISCRETE_DERIVED_VALUE
– EGSE_STRING_DERIVED_VALUE

The different types of Derived Values are defined according to the data type of the respective engineering value.

End items declared as Derived Values can be referred to in UCL APs, procedures or functions, HLCL sequences, synoptic displays, etc. via their pathname.

The overall aggregate structure of all Derived Values end items in MDB is the same and comprises:

- an Engineering Value Description which is mandatory for the integer and float Derived Values, but not required for the discrete ones.
- The source code of an expression in UCL syntax specifying the calculation of the value
- The compiled items for the UCL expression, as there are
  – the I–Code
  – the symbol tables
  – the cross reference table
  – the compilation date and the compilation status
- an Engineering Value Logging flag which is optional. It indicates whether or not the value shall be logged.
- a Monitoring Definition which is optional. This aggregate defines the limits against which the actual value of the SW variable shall be checked and which actions shall be taken in case the limits are violated (out of limit situation)
- an optional Condition Definition. This aggregate defines conditions for the enditem, which control the processing or monitoring of other enditems or the starting of APs.

The source code of the expression must follow the following rules:

– UCL syntax is applicable (refer to UCL LRM). The following subset of statements is allowed:
  – import statement
  – if statement
  – case statement
  – return statement
  – function calls
  (to standard functions/procedures, system libraries and user libraries)
  – use of unitized values
  – type conversions and call to standard functions

– The following is not allowed:
  –no declarations

–no assignments
– import allowed only for system libraries, not for user libraries
– Only the following declarations are possible:
  — Constants
  — Type declarations
  — Alias
– no procedure calls
– No assignments to variables or MDB Objects

Note: The restrictions are introduced to avoid overloading of the monitoring function with UCL processing and to avoid execution of any library procedure (e.g. sending a command, disable archiving etc.) from within derived value expressions.

Example of Expression for an Integer_Derived_Value:

```
return \satellite\ss\active_count + \satellite\boxes\active_count;
```

Example for Float_Derived_Value:

```
if \satellite\powerstatus = $ON
   then
      return \sat\subsys_a\power\current + \sat\subsys_a\power\current;
   else
      return 0.0 [A];
end if;
```

Example of Expression for a Discrete_Derived_Value:

```
if \satellite\powerstatus = $ON
 then return $POWER;
 else return $UNDEFINED;
end if;
```

Example of Expression for a String_Derived_Value:

```
if \satellite\powerstatus = $ON
 then return ”Power is active”;
 else return ”Power is inactive”;
end if;
```

7.6.3.3.1 EGSE_INTEGER_DERIVED_VALUE

This type describes Derived Value end items with *integer* engineering values.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
Expression description		
EXPRESSION SOURCE	empty	mandatory A UCL expression as defined above Return statement must return IN-TEGER type Must be compiled/stored (Menu: Tools–> CLS_Editor). Then the created I–Code is stored in addi-tional aggregates of the MDB.
engineering value description		
INTEGER ENGINEERING RANGE	MIN_INTEGER .. MAX_INTEGER(*)	optional
EVL CONTROL	FALSE(*)	Activates logging of variable in online checkout.
Monitoring Description		Defines limits and actions for nominal monitoring and danger monitoring. Note: If danger limits are defined, nominal limits must be defined too.
INTEGER NOMINAL LIMITS All 5 limit sets Nominal_Low_Action Nominal_High_Action Nominal_Delta_Action Nominal_Low_Message Nominal_High_Message Nominal_Delta_Message	 <empty> (*) <empty> (*) <empty> (*)	optional If any limit is defined, high and low limits must be defined Delta limits are optional. If any, limit set 1 must be defined. AP or STIMULUS/PREDEFINED_TC/ BINARY_PACKET maybe specified by Pathname Additional message text which is dis-played in the message window when an limit violation occurs, specified by pathname

Aggregates	Standard Value / Default (*)	Comment
INTEGER DANGER LIMITS Low Limit,High Limit,Delta Limit Danger_Low_Action Danger_High_Action Danger_Delta_Action Danger_Low_Message Danger_High_Message Danger_Delta_Message	 <empty> (*) <empty> (*) <empty> (*)	optional If any limit is defined, high and low limits must be defined Delta limits are optional. AP or STIMULUS/PREDEFINED_TC/ BINARY_PACKET maybe specified by Pathname Additional message text which is displayed in the message window when an limit violation occurs, specified by pathname
MAX ALARM COUNTER	1 (*)	MAX ALARM COUNTER = n means that after n times of consecutive limit violations the exception for nominal limits will be raised
Conditions		
INTEGER_CONDITIONS	empty	optional Refer to EGSE_INTEGER_MEASUREMENT

Consistency Checks:

Danger High Limit <= High Value in the engineering description

Danger Low Limit >= Low Value of the engineering description

The referenced AP action in monitoring limits or conditions must not have parameters.

If the referenced stimulus/TC/BP action in monitoring limits has parameters, default values must be assigned to each. Default values will be used when they are sent.

Low Limit < High Limit

If Action = ’Switch_Limit_Set’, the referenced enditem must have nominal limit sets defined

If Action = ’Switch_Limit_Set’, the given Limit Set Number must be defined for the Action_Enditem_Reference

The Action_Enditem_Reference must be within the allocated set of enditems for the test node, where the enditem carrying the condition is allocated to.

Notes: For notes on conditions refer to EGSE_INTEGER_MEASUREMENT

7.6.3.3.2 EGSE_FLOAT_DERIVED_VALUE

This type describes Derived Value end items with *float* engineering values.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
Expression description		
EXPRESSION SOURCE	empty	mandatory A UCL expression as defined above Return statement must return REAL type Must be compiled/stored (Menu: Tools–> CLS_Editor). Then the created I–Code is stored in additional aggregates of the MDB.
engineering value description		
ENGINEERING UNITS		optional
FLOAT ENGINEERING RANGE	MIN_FLOAT .. MAX_FLOAT (*)	optional
EVL CONTROL	FALSE (*)	Activates logging of measurement in online checkout.
Monitoring Description		Defines limits and actions for nominal monitoring and danger monitoring. Note: If danger limits are defined, nominal limits must be defined too.
FLOAT NOMINAL LIMITS All 5 limit sets Nominal_Low_Action Nominal_High_Action Nominal_Delta_Action Nominal_Low_Message Nominal_High_Message Nominal_Delta_Message	 <empty> <empty> <empty>	optional If any limit is defined, high and low limits must be defined Delta limits are optional. If any, limit set 1 must be defined. AP or STIMULUS/PREDEFINED_TC/BINARY_PACKET maybe specified by Pathname Additional message text which is displayed in the message window when an limit violation occurs, may be specified by pathname

Aggregates	Standard Value / Default (*)	Comment
FLOAT DANGER LIMITS Low Limit,High Limit,Delta Limit Danger_Low_Action Danger_High_Action Danger_Delta_Action Danger_Low_Message Danger_High_Message Danger_Delta_Message	 <empty> (*) <empty> (*) <empty> (*)	optional If any limit is defined, high and low limits must be defined Delta limits are optional. AP or STIMULUS/PREDEFINED_TC/ BINARY_PACKET maybe specified by Pathname Additional message text which is displayed in the message window when an limit violation occurs, may be specified by pathname
MAX ALARM COUNTER	1 (*)	MAX ALARM COUNTER = n means that after n times of consecutive limit violations the exception for nominal limits will be raised
Conditions		
FLOAT_CONDITIONS	empty	optional Refer to EGSE_FLOAT_MEASUREMENT

Consistency Checks:

Danger High Limit <= High Value in the engineering description

Danger Low Limit => Low Value of the engineering description

The referenced AP action in monitoring limits must not have parameters.

If the referenced stimulus/TC/BP action in monitoring limits has parameters, default values must be assigned to each. Default values will be used when they are sent.

Low Limit < High Limit

If Action = ’Switch_Limit_Set’, the referenced enditem must have nominal limit sets defined

If Action = ’Switch_Limit_Set’, the given Limit Set Number must be defined for the Action_Enditem_Reference

The Action_Enditem_Reference must be within the allocated set of enditems for the test node, where the enditem carrying the condition is allocated to.

Notes: For notes on conditions refer to EGSE_INTEGER_MEASUREMENT

7.6.3.3.3 EGSE_DISCRETE_DERIVED_VALUE

This enditem type describes a SW_VARIABLE with a *state code* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an AP.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
Expression description		
EXPRESSION SOURCE	empty	mandatory A UCL expression as defined above Return statement must return STATE_CODE type Must be compiled/stored (Menu: Tools–> CLS_Editor). Then the created I–Code is stored in additional aggregates of the MDB.
engineering value description		
EVL CONTROL	FALSE(*)	Activates logging of measurement in online checkout.Initially, engineering value logging should be disabled
discrete calibration		for use by GWDU only !
DISCRETE_CALIBRATION up to 256 Statecode / Raw Value Specifications. A Raw Value Spec. is a single value or a range: Raw_Value..High_Value_of_Range	if empty, the OTHERS statecode applies for any raw value	optional Statecodes are strings up to 8 characters, starting with a letter and containing letters/digits and underscores Raw values are not applicable for Derived Values
UNDEFINED_VALUES_STATE_CODE	OTHERS (*)	optional N/A for Derived Values
Monitoring Description		
DISCRETE MONITOR LIST Up to 5 Limit Sets Expected Value Exception Message Exception Action	empty (*)	one of the state codes defined in the DISCRETE CALIBRATION Expected Statecode/String Message to be generated if expected value is not the same as the actual value GDU/AP to be sent/started if expected value is not the same as the actual value
MAX ALARM COUNTER	1 (*)	MAX ALARM COUNTER = n means that after n times of consecutive expected value violations the exception for nominal limits will be raised

Aggregates	Standard Value / Default (*)	Comment
Conditions		
DISCRETE_CONDITIONS	empty	optional Refer to EGSE_DISCRETE_MEASUREMENT

Consistency Checks:

The referenced AP action in monitoring limits or conditions must not have parameters.

If the referenced stimulus/TC/BP action in monitoring limits has parameters, default values must be assigned to each. Default values will be used when they are sent.

If discrete conditions are specified, the expected string must be defined in the discrete calibration

If Action_Type = 'Switch_Limit_Set', the referenced enditem must have nominal limit sets defined

If Action_Type = 'Switch_Limit_Set', the given Limit Set Number must be defined for the Action_Enditem_Reference

The Action_Enditem_Reference must be within the allocated set of enditems for the test node, where the enditem carrying the condition is allocated to.

Notes: For notes on conditions refer to EGSE_INTEGER_MEASUREMENT

7.6.3.3.4 EGSE_STRING_DERIVED_VALUE

This enditem type describes a Derived Value with an engineering value of type *string*.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
Expression description		
EXPRESSION SOURCE	empty	mandatory A UCL expression as defined above Return statement must return STRING type Must be compiled/stored (Menu: Tools–> CLS_Editor). Then the created I–Code is stored in additional aggregates of the MDB.
engineering value description		
EVL CONTROL	FALSE(*)	Activates logging of measurement in online checkout.
Monitoring Description		
BYTESTREAM MONITOR LIST Up to 5 Limit Sets: Expected Value Exception Message Exception Action	empty (*)	 Expected String Message to be generated if expected value is not the same as the actual value GDU/AP to be sent/started if expected value is not the same as the actual value
MAX ALARM COUNTER	1 (*)	MAX ALARM COUNTER = n means that after n times of consecutive expected value violations the exception for nominal limits will be raised
Conditions		
BYTE_STREAM_CONDITIONS	empty	optional Refer to EGSE_BYTE_STREAM_MEASUREMENT

Consistency Checks:

The referenced AP action in monitoring limits must not have parameters.

If the referenced stimulus/TC/BP action in monitoring limits has parameters, default values must be assigned to each. Default values will be used when they are sent.

If Action = ’Switch_Limit_Set’, the referenced enditem must have nominal limit sets defined

If Action = ’Switch_Limit_Set’, the given Limit Set Number must be defined for the Action_Enditem_Reference

The Action_Enditem_Reference must be within the allocated set of enditems for the test node, where the enditem carrying the condition is allocated to.

Notes: For notes on conditions refer to EGSE_INTEGER_MEASUREMENT

7.6.3.4 Messages

User messages are a convenient mechanism to provide more useful information to the user in case of erroneous or abnormal situations, e.g. when monitoring the incoming data from the unit under test.

EGSE_USER_MESSAGES can be refered to in monitoring definitions of measurements / SW Variables (Attribute ”Exception Message”). The message is then sent to the Message Window and to the Event Log in case of limit violations.

7.6.3.4.1 EGSE_USER_MESSAGE

The formal definition is:

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
EGSE MESSAGE SHORT_TEXT LONG_TEXT	 empty (*) empty (*)	Strings are sent as message to the CGS message window – if referenced in an limit violation – when violation occurs String appearing in the message window overview resp. as Short_Text in the event log String appearing in the message window supplement window (when double clicking on the overview line) resp. as Long_Text in the event log
FORMAL_PARAMETER_SOURCES		not used

7.6.3.5 Test Facility Description

There are a number of database enditems which describe the test facility and the configuration of the facility for a given test. This includes definitions of the individual HW nodes of the test equipment, the individual SW entities and the overall configuration. All items can be configured to a test configuration which can be selected during test setup and activated for a given test. This test configuration defines then the participating nodes, the needed SW entities , the CCU to be applied as well as individual items to be loaded to the test nodes.

Note: For a defined Test Configuration, the Tool “Generate_Scoe_Files” must be called within a slected CCU, whenever items allocated to the Test Configuration have been changed.

7.6.3.5.1 EGSE_NODE

This enditem describes a specific part of the HW of the facility. Its main purpose is to provide access to parts of the facility via pathnames from UCL/HLCL level. It also defines the ’role’ of the node, e.g. in the EGSE environment it determines whether the node is a workstation, a test node, a database server, a simulation node, a front–end equipment or a UUT. Finally, configuration management of this piece of the facility can be performed using the standard MDB Configuration Management provisions and the Test Setup and Configuration (TSCV) program.

Aggregates	Standard Value / Default (*)	Comment
NODE_TYPE	TEST_NODE	mandatory WORKSTATION: the node runs the user interface of CGS: HCI, TEV, MOCS etc. DATABASE_SERVER: the node runs ORACLE Processes, the DBS Processes and the FA_SAS TEST_NODE: The node runs the TES process and associated SAS SIMULATION_NODE: The node runs the CSS Kernel and the CMAS Software FRONT_END_EQUIPMENT: Node runs non–CGS software and is connected as a frontend of the test node or simulation node UNIT_UNDER_TEST: Node is part of the unit under test
LOGICAL_NAME	empty	optional (not used for test configurations) mandatory for node types FRONT_END_EQUIPMENT and UNIT_UNDER_TEST
CGS_INTERNAL_NAME	CGS_PREFIX: TES INTERNAL_NUMBER: 1	mandatory CGS_PREFIX TES: for test nodes DBS: for DATABASE_SERVER HCI: For workstations CSS: for Simulation Nodes INTERNAL_NUMBER 1 for DBS 1..32 for test nodes 1..32 for simulation nodes (Currently only 1 is supported) 1..32 for HCI optional for node types FRONT_END_EQUIPMENT and UNIT_UNDER_TEST

Consistency Checks:

CGS_INTERNAL_NAME must be unique for all EGSE NODES

7.6.3.5.2 EGSE_SOFTWARE

This enditem describes a specific part of the SW of the facility. Its main purpose is to provide access to this SW via pathname from UCL/HLCL level. It also defines the 'type' of the SW, e.g. in the EGSE environment it determines whether the SW is a SAS, a data file, etc. Finally, configuration management of this piece of the facility can be performed using the standard provisions of the MDB

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
EGSE SOFTWARE		
SW_TYPE	SAS	mandatory SAS The software is a SAS running on test nodes EXECUTABLE The software is an executable (but no SAS) DATA_FILE The software is a data file
FILE_NAME	$GSAF_HOME/sas/bin (*)	optional (not used for test configurations) UNIX File Name Note: SAS must be located under $GSAF_HOME/sas/bin. The file-name given here is not relevant for finding the SAS in TES.
SWEU_REFERENCE	empty	optional (not used for test configurations) Pathname of SWEU
SHORT_NAME	empty	mandatory if SW_TYPE = SAS The name of the SAS used in UCL LOAD command to address the SAS (up to 20 character) For SAS, it must be identical to name of UNIX executable and it must be unique within a CCU No space (blank) character allowed Note: The short name of the SAS must also be used within the System_Topology_Table when defining port numbers and the physical node (UNIX node name) where the SAS shall run

7.6.3.5.3 EGSE_TEST_CONFIGURATION

This enditem describes the actual configuration of the facility for a given checkout application. It is the actual configuration of the test equipment to be set–up for a given test.

The test configuration is described in terms of references to other MDB items of type EGSE_NODE and EGSE_SW plus the definition of MDB contents (CDUs) to be 'downloaded' to test nodes. For test nodes, the role of the Node (being the Master Test Processor or ordinary test node) , the execution mode and predefined items such as the AP to be executed when started in batch mode, or the allocated overview synoptic can be defined.

For test nodes and workstations, a flag indicates, if the node is foreseen to participate. This flag can be modified during the setup in TSCV.

Enditems of type test configuration are later used in the CGS product TSCV to actually perform the set-up. Several Test Configurations may be defined. Up to 5 can be loaded and activated in parallel, given that their nodes do not participate in more than one of them.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
DATABASE NODE	empty	mandatory Must be a pathname to an EGSE_NODE having DATABASE_SERVER as node type
SIMULATOR NODE	empty	Optional Must be a pathname to an EGSE_NODE having SIMULATION_NODE as node type
EGSE TEST NODES		
TEST_NODE	empty	Must be a pathname to an EGSE_NODE having TEST_NODE as node type
IS_MASTER_TEST_PROCESSOR	FALSE	TRUE/FALSE Defines the MTP role of the test node (One and only one MTP can exist in a Test Configuration)
INITIAL_AUTOMATED_PROCEDURE	empty (*)	AP which is executed when test node is set into RUNNING state. (In TSCV Batch mode only)
IS_PARTICIPATING	TRUE	TRUE/FALSE Defines if the node is participating in the test configuration (If false, the node may be switched to participating later via TSCV, after load of the test configuration)

Aggregates	Standard Value / Default (*)	Comment
EXECUTION_MODE	NORMAL	NORMAL,SIMULATION,RE-PLAY Mode of the test node NORMAL: normal checkout operation, where data is acquired and sent to frontends via SAS SIMULATION: test nodes are simulating ADUs and do not sent GDUs to SAS. REPLAY: data archived in previous test sessions is replayed
OVERVIEW_SYNOPTIC	empty (*)	Any synoptic; should give overview picture on the data processed by the test node, but no predefined restrictions exist. Synoptic is automatically called when Subsystem Name is selected in the System Advisory Window of HCI
SUBSYSTEM_NAME	empty	Name of the Subsystem allocated to the test node Will appear in the System Advisory window of HCI Name can freely be chosen, but should express the subsystem monitored by the test node
EGSE_WS_NODES		
WORKSTATION_NODE	empty	Must be a pathname to an EGSE_NODE having WORKSTATION as node type
IS_PARTICIPATING	TRUE	TRUE/FALSE Defines if the node is participating in the test configuration (If false, the node may be switched to participating later via TSCV, after load of the test configuration)

Aggregates	Standard Value / Default (*)	Comment
EGSE TEST NODE_CDUs		
TEST_NODE	empty	Denotes the test node where the CDU shall be allocated/loaded to. Must be a pathname to an EGSE_NODE having TEST_NODE as node type;
LOADED_ITEMS	empty	Items of the MDB which shall be loaded to the Test Node. All Items must also be visible in the CCU that is used for testing. Items may be – complete CDUs – virtual pathes – single enditems Note: Check that really the same versions of the CDUs are referenced here as they are included in the CCU! Measurements referenced in ADUs cannot be allocated directly. Their allocation is derived from the ADUs. Therefore only the ADUs must be allocated. Virtual pathes/CDUs shall not overlap eachother. The data is stored via the Tools–>Generate_SCOE_Files Menu of the Test Configuration into files and then later loaded by the test nodes (TES) when going to the IDLE/RUNNING state (upon INIT command in the Test System Setup function) At least one item must be defined for each test node
EGSE TEST NODE_SASs		
TEST_NODE	empty	Denotes the test node where the SAS shall be allocated/loaded to. Must be a pathname to an EGSE_NODE having TEST_NODE as node type;
SAS	empty	SAS which shall be running on the Test Node. A SAS is defined as an enditem of type EGSE_SOFTWARE. Up to 20 SAS may be specified Each SAS may appear only once in the list

Consistency Checks:

(1) If specified, the pathname must refer to an existing MDB item of type:
- CDU
- VIRTUAL
- UNSTRUCTURED_ADU_DESCRIPTION
- STRUCTURED_ADU_DESCRIPTION
- CCSDS_ADU_DESCRIPTION
- EGSE_xxx_DERIVED_VALUE
- EGSE_INTEGER_SW_VARIABLE
- EGSE_FLOAT_SW_VARIABLE
- EGSE_DISCRETE_SW_VARIABLE
- EGSE_BYTE_STREAM_SW_VARIABLE
- EGSE_ANALOG_STIMULUS,
- EGSE_DISCRETE_STIMULUS,
- EGSE_PREDEFINED_TC,
- EGSE_BINARY_PACKET
- EGSE_MONITOR_LIST
- GDU_DESCRIPTION_LIST
- SIMULATED_ADU_DESCRIPTION
- EGSE_SOFTWARE
- EGSE_USER_MESSAGE
- UCL_USER_LIBRARY
- RESPONSE_PACKET
- APID
- SWOP_COMMAND

(2) If the given items appears in more than one Loaded Item List (i.e. if the items are loaded into more than one test node), then this items may not of type:
- EGSE_xxx_SW_VARIABLE
- EGSE_xxx_DERIVED_VALUE
- UNSTRUCTURED_ADU_DESCRIPTION
- STRUCTURED_ADU_DESCRIPTION
- CCSDS_ADU_DESCRIPTION
- EGSE_ANALOG_STIMULUS
- EGSE_DISCRETE_STIMULUS
- EGSE_PREDEFINED_TC
- EGSE_BINARY_PACKET
- EGSE_MONITOR_LIST
- GDU_DESCRIPTION_LIST

(3) The referenced items must belong to the same CCU as the EGSE Test Configuration end item being defined.

(4) At least one item must be downloaded into each test node.

(5) The list of items loaded for one test node must contain all end items of type EGSE_xxx_STIMULUS, EGSE_PREDEFINED_TC or EGSE_BINARY_PACKET which are referenced in all end items of type GDU_DESCRIPTION_LIST in this list of items.

(6) The SASes referenced in all EGSE_xxx_STIMULUS, EGSE_PREDEFINED_TC, EGSE_BINARY_PACKET and xxx_ADU_DESCRIPTION end items contained in all the specified items for one test node shall also exist in the list of ”Used SAS” (see below).

(7) The xxx_ADU_DESCRIPTIONs referenced in all SIMULATED_ADU_DESCRIPTION end items contained in all the specified items for one test node shall exist within the same scope.

(8) The list of items loaded for one test node must contain all end items of type EGSE_SW_xxx_VARIABLE or EGSE_xxx_DERIVED_VALUE which are referenced in all end items of type MONITOR_LIST and in this list of items.

(9) The EGSE_xxx_MEASUREMENTs referenced in all MONITOR_LIST end items contained in all the specified items for one test node shall also referenced in xxx_ADU_DESCRIPTION end items contained in the same scope.

(10) All xxx_ADU_DESCRIPTIONs which are references to the same EGSE_xxx_MEASUREMENT end item shall exist in the same scope of specified items for one test node.

(11) The UCL_USER_LIBRARY, EGSE_MONITOR_LIST, GDU_DESCRIPTION_LIST, EGSE_NODE, EGSE_SOFTWARE, EGSE_USER_MESSAGE, xxx_ADU_DESCRIPTION and VIRTUAL end items referenced in all UCL_AUTOMATED_PROCEDURE end items contained in all the specified items for one test node shall exist within the same scope.

(12) The UCL_AUTOMATED_PROCEDURE,   EGSE_xxx_SW_VARIABLE, EGSE_xxx_DERIVED_VALUE, EGSE_xxx_STIMULUS, EGSE_BINARY_PACKET and EGSE_PREDEFINED_TC end items referenced in all UCL_AUTOMATED_PROCEDURE end items contained in all the specified items for a whole test configuration shall exist within the same scope.

(13) The EGSE_xxx_MEASUREMENT end items referenced in UCL_AUTOMATED_PROCEDURE end items contained in all the specified items for a whole test configuration shall be referenced at least in one xxx_ADU_DESCRIPTION of the same scope.

(14) For SWOP_COMMANDS referenced in UCL_AUTOMATED_PROCEDURE end items contained in all the specified items for a whole test configuration the aggregate T_STIMU-LUS_GENERAL_INFO is mandatory.
It must have valid values for the following attributes:

SAS_REFERENCE

(15) The end items referenced in EGSE_xxx_STIMULUS, EGSE_PREDEFINED_TC, EGSE_BINARY_PACKET, EGSE_xxx_DERIVED_VALUE, EGSE_xxx_MEASUREMENT and EGSE_xxx_SW_VARIABLE enditems contained in all the specified items for a whole test configuration shall exist within the same scope.

(16) No cycle shall exist within the references of EGSE_xxx_DERIVED_VALUE enditems contained in all the specified items for a whole test configuration

7.6.3.6 Stimuli

In the checkout environment, the stimuli enditems describe commands to be sent to the unit under test, front–end equipment or a SAS via the generic output data packet GDU. GDU stands for ”Generation Data Unit” and is another word for a data block sent to SAS. The internal structure of this data block needs not to be known when filling the MDB with data. However, the purpose of the stimulus has to be known, i.e. whether it is meant for the SAS, a front end or the unit under test and how it shall be sent (via a dedicated HW interface, a front–end equipment or the ground to space link. Thus, stimuli include telecommands sent via CCSDS packets or other telecommand formats.

Several types of stimuli and corresponding MDB enditems are available, being

– EGSE_ANALOG_STIMULUS
– EGSE_DISCRETE_STIMULUS
– EGSE_PREDEFINED_TELECOMMAND
– EGSE_BINARY_PACKET

– INTEGER_STIMULUS
– UNSIGNED_INTEGER_STIMULUS
– DOUBLE_FLOAT_STIMULUS
– BOOLEAN_STIMULUS
– PULSE_STIMULUS
– BURST_PULSE_STIMULUS
– SIM_TC_PACKET_DESCR(IPTION)

The different types of stimuli are defined according to the type of data associated with it, i.e. an analog stimulus comes along with an analog (float) value (e.g. a command to set the output voltage of a power supply to 28.0 Volt) a discrete stimulus has a discrete value (e.g. switch a pump ON) or a predefined TC has a bit pattern associated which is the telecommand to be sent (e.g. a CCSDS telecommand).

Stimuli are ”parameterized” items. Their type definitions comprise a formal parameter list describing the type of values that must (or may) be supplied at runtime.

Enditems declared as stimuli can be referred to in UCL APs, procedures or functions, HLCL sequences, synoptic displays, etc. via their pathname and then result in the defined command being sent to SAS.

Enditems declared as stimuli can also be referred to in monitoring definitions as actions, meaning that they are sent to the SAS when the out of limit condition is reached.

In simulation models, stimuli are used to describe input/output values of the models. The may be received from external sources (CMAS and front ends)

In checkout environments (VICOS), all stimuli/telecommnds/packets are refered to as Generation Data Units (GDU).

The overall property structure of all stimuli enditems in MDB is comprises:

- a *general description* which is **mandatory**. This description contains some identification information which is needed for CGS and the SAS to properly handle the command.
- a *physical address*, which is **optiona**l. This aggregate contains information needed by the SAS/CMAS to physically process the command and is not processed by CGS.
- a *raw value description* which is **optional**. This description is not needed for checkout but for simulation purposes only.
- an *engineering value description* which is **optional**. This description is not needed for checkout but for simulation purposes only.
- a *parameter list* which is **optional**. If present, 1 to 255 elements can be defined. The parameters are substituted into the GDU actually generated at run–time. Only a constraint number of data types is allowed as parameter.
- for each parameter, a *de–calibration description* which is **optiona**l.

  In the checkout environment, the decalibration allows to specify engineering units which are converted/decalibrated to raw values before they are sent / inserted into the packet/GDU. For simulation models, the stimuli are received.Therefore their values are calibrated.
- a *Command Verification Description*, which is **optional**

  In the checkout environment, a stimulus/TC Packet is sent and afterwards, a verification may be performed by checking that referenced measurements/sw variables/derived values have a specified value.

  The descriptions includes two times:

  Activation_Delay_in_Seconds
  A float value describing the number of seconds to wait before the verification of the measurements is started

  Verification_Timeout_in_Seconds
  A float value describing the number of seconds to wait after the ACTIVATION_DELAY has expired and before the verification of the measurements is finished

7.6.3.6.1 EGSE_ANALOG_STIMULUS

This enditem describes a command to be sent to a SAS and/or a front end equipment and/or the unit under test comprising one analog value as a parameter.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
STIMULUS GENERAL INFO		General Attributes
Private_ID	empty string (*)	Foreseen for SAS, must be unique within CCU
Number_of_Retries	1 (*)	Numer of retries in sending to the SAS
SAS_Reference		pathname of SAS (EGSE_SW) which the stimulus is to be sent to
Inhibited_for_sending	false (*)	Stimuli/GDU may be inhibted for sending. Only dangerous commands should be inhibited (can be enabled by specific UCL command)
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical address where to sent the stimulus/GDU to
Engineering Value Description		Engineering value of Parameter
FLOAT ENGINEERING RANGE	MIN_FLOAT .. MAX_FLOAT	Range of Engineering Value
ENGINEERING_UNITS		Engineering Unit for Parameter; optional;
Raw Value Description		Raw value of Parameter
RAW_VALUE_TYPE	FLOAT	Specifies type of raw value where the parameter is decalibrated to Only FLOAT is supported by VICOS.
RAW VALUE SIZE IN BITS	32	Size of the raw value; Only 32 allowed
FLOAT_RAW_VALUE_RANGE	MIN_FLOAT .. MAX_FLOAT	Range of Raw Value

Aggregates	Standard Value / Default (*)	Comment
Decalibration/Calibration		Describes decalibration (for VICOS) resp. calibration (for CSS)
CURVE TYPE	POLYNOM	mandatory IDENTICAL,POLYNOM or POINT_PAIRS For VICOS: IDENTICAL means same as POLYNOM with coefficients 0.0,1.0,0.0,0.0,0.0
ANALOG POINT PAIRS	(0,0) (MAX_UNSIGNED, MAX_UNSIGNED)	mandatory (if curve type = point pairs) used for CSS/CMAS only
ANALOG_DECAL_COEFFICIENTS	0.0, 1.0, 0.0, 0.0, 0.0	mandatory (if curve type = polynom) used for VICOS only
ANALOG DECAL POINT PAIRS	(MIN_FLOAT,MIN_FLOAT) (MAX_FLOAT,MAX_FLOAT)	optional used for VICOS only
Formal Parameter Definition		Formal Definition of the parameter
FORMAL PARAMETER SOURCES	(PARAM: REAL)	Defines in UCL syntax the definition of the parameter as well as its default value Only REAL is allowed Must be compiled/stored via the CLS Editor. Then the definition of the parameter is stored in other aggregates of the MDB.
Verification Description		
COMMAND_VERIFICATION TIMES Activation_Delay_in_Seconds Verification_Timeout_in_Secs	 0.0 0.0	Times to wait before verification starts after sending A float value describing the number of seconds to wait before the verification of the measurements is started If this value is set to 0.0, the checking shall be performed immediately after the command has been acknowledged. A float value describing the number of seconds to wait after the Activation_Delay has expired and before the verification of the measurements is finished If this value is set to 0.0, the checking shall be performed immediately after the Activation_Delay and then be finished.

Aggregates	Standard Value / Default (*)	Comment
COMMAND_VERIFICATION		Defines the enditems to be checked and the values to be checked acc to the expression: Measurement Operator Value [High_Value_for_range] e.g.: ”\a\b\c in_range 10 20”
Measurement_to_be_checked	(undefined)	A measurement which is subject to be verified after the command has been issued. Must be of type EGSE_..._MEASUREMENT, EGSE_..._SW_VARIABLE or EGSE_..._DERIVED_VALUE
Operator	(undefined)	>, >=, <, <=, =, /=, in_range If Measurement_to_be_checked is of type EGSE_DISCRETE_... , EGSE_BYTE_STREAM_... or EGSE_STRING_..., Operator must be one of = or /=
Value(Low_Value for range)		Expected Value of Measurement If Operator = in_range: Low Value of Range
High_Value for range		If Operator in_range: High Value of Range
Command Authorization		
CRITICAL_COMMAND	false (*)	true / false If “true”, the command is to be authorized via password before sending
PASSWORD	empty	String up to 8 character Defines the password which has to be entered before sending

Consistency Checks:

Only one Formal Parameter allowed

Formal Parameter's mode must be: IN

Formal Parameter's type must be: REAL_TYPE

If Measurement_to_be_checked is of type EGSE_FLOAT_... , strings given in Value and High_Value_for_Range must be convertable to a float value.

If Measurement_to_be_checked is of type EGSE_INTEGER_..., strings given in Value and High_Value_for_Range must be convertable to an integer value.

If Measurement_to_be_checked is of type EGSE_DISCRETE_..., string given in Value must be limited to 8 upper–case characters (state_code).

If Operator is "in_range", both Value and High_Value_for_Range must ge given

If Operator is "in_range", Measurement_to_be_checked must of type EGSE_FLOAT_... or EGSE_INTEGER_...

7.6.3.6.2 EGSE_DISCRETE_STIMULUS

This enditem describes a command to be sent to a SAS and/or a front end equipment and/or the unit under test comprising one discrete value as a parameter. This end item is also the counterpart to the CSS top level I/O of type state code.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
STIMULUS GENERAL INFO (refer to EGSE_ANALOG_STIMULUS)		General Attributes
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical address where to sent the stimulus/GDU to
Raw Value Description		Raw value of Parameter
RAW_VALUE_TYPE	UNSIGNED_INTEGER	Specifies type of raw value where the parameter is decalibrated to Only UNSIGNED_INTEGER is supported by VICOS.
RAW VALUE SIZE IN BITS	31	Size of the raw value; 1..31
UNSINGED_INT_RAW_VALUE RANGE	0.. MAX_UNSIGNED	Range of Raw Value
Decalibration/Calibration		Describes decalibration (for VICOS) resp. calibration (for CSS)
DISCRETE_CALIBRATION up to 31 Statecode / Raw Value Pairs		Used for decalibration in VICOS Used for calibration in CSS/CMAS Statecodes are strings up to 8 characters The statecode ”OTHER” is reserved and must not be used.
UNDEFINED_VALUES_STATE_CODE	OTHERS (*)	optional Defines statecode for all raw values which do not appear in the list of 31 statecodes above Not used for VICOS
Formal Parameter Definition		Formal Definition of the parameter
FORMAL PARAMETER SOURCES	(PARAM: STATECODE)	Defines in UCL syntax the definition of the parameter as well as its default value Only STATE_CODE is allowed. Must be compiled/stored via the CLS Editor. Then the definition of the parameter is stored in other aggregates of the MDB.

Aggregates	Standard Value / Default (*)	Comment
Verification Description		
COMMAND_VERIFICATION TIMES		Times to wait before verification starts after sending See EGSE_ANALOG_STIMULUS
COMMAND_VERIFICATION		Defines the enditems to be checked and the values to be checked See EGSE_ANALOG_STIMULUS
Command Authorization		
CRITICAL_COMMAND	false (*)	true / false If “true”, the command is to be authorized via password before sending
PASSWORD	empty	String up to 8 character Defines the password which has to be entered before sending

Consistency Checks:

Only one Formal Parameter allowed

Formal Parameter’s mode must be: IN

Formal Parameter’s type must be: STATE_CODE_TYPE

For checks on COMMAND_VERIFICATION see EGSE_ANALOG_STIMULUS

7.6.3.6.3 EGSE_BINARY_PACKET

This is a parameterizable MDB end–item defining a packet of data items sent within a data buffer.

This end–item type defines a stimulus having a stream of binary data as contents. The enditem is sent to an SAS within a GDU. It is a *parameterized* end item type with up to 255 parameters. The data stream associated with the enditem may consist of up to 4096 bytes. Similar to the other stimulus definitions, the enditem definition contains some general information, physical address information and a parameter list. In addition, however, the necessary data to describe the layout of the binary packet has to be provided in the following aggregates:

- Bit stream layout description. This aggregate describes the predefined static bit patterns used for initializing the binary packet, i.e. the byte stream associated with it.
- a *Command Verification Description*, which is **optional**

  See EGSE_ANALOG_STIMULUS
- Formal parameter list (up to 255 parameters are allowed).
  For each parameter:
  - an optional raw value description.
  - an optional engineering value description.
  - an optional decalibration description. Depending on the parameter type, this may be analog (INTEGER, REAL) or discrete (STATE_CODE) decalibration.

Note: Parameter are specified by the CLS Editor as usual, e.g. :
(PARA1: INTEGER := 3.14);

The meaning of the formal parameter list defined with this enditem is somehow different from the one defined for the other stimuli enditems. Here, the formal parameters defined for the binary packet will be substituted at run–time into the data buffer by CGS. The logic with respect to the handling of the bitstream layout and parameters is that first the static bit patterns defined by bitstream layout is put into the data buffer and then the actual values of the formal parameters will be substituted, possibly overwriting the static bits.

Note: Binary Packets are issued via the UCL Issue command, as in:
ISSUE (\A\B\C (123, 1.0, 100, $ON), stat);

The enditem is only used for checkout (VICOS).

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
STIMULUS GENERAL INFO (refer to EGSE_ANALOG_STIMULUS)		General Attributes
GLOBAL_LENGTH	4096 (*)	Length (in bytes) of data part of packet Maximum: 4096
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical address ehwere to sent the stimulus/GDU to
Bitstream Layout		Definition of predefined bitstreams (data) for the binary packet
GENERAL_BITSTREAM_LAYOUT Bitstream:Position Bitstream:Format Bitstream:Location		A list of up to 255 bitstream items which shall be inserted into the packet as predefined values. For each item a corresponding BINARY/FLOAT/INTEGER/UNSIGNED DEFINITION must exist Identifies the position in the list INTEGER, UNSIGNED, FLOAT or BINARY Identifies the location of the bitstream item within the data part of the packet 1..8*length of data part
INTEGER_DEFINITION Position Value Number of bits 32	 32	 References the item in the List of Bitstream Items Value which is stored in the TC data buffer at the location specified in the referenced bitstream item Number of bits the item occupies. 1..32
UNSIGNED_INTEGER_DEFINITION Position Value Number of bits 32	 32	 References the item in the List of Bitstream Items Value which is stored in the TC data buffer at the location specified in the referenced bitstream item Number of bits the item occupies. 1..32

Aggregates	Standard Value / Default (*)	Comment
FLOAT_DEFINITION Position Value		 References the item in the List of Bitstream Items Value which is stored in the TC data buffer at the location specified in the referenced bitstream item. Is stored as 32 bit floating point value (IEEE format)
BINARY_DEFINITION Position Value Type Value		 References the item in the List of Bitstream Items ASCII: give value as an ASCII string other: HEX: give value as an hexadecimal number string(1..255) either the string value itself or the Hexadecimal Representation of the value The value is stored in the packet data buffer at the location specified in the referenced item The length of the value is determined by the string value resp. by the hexadecimal value Note: Hex values must be specified for full bytes amd always with 2 characters for each byte (e.g. ”0F”, not ”F”)
Parameter Raw Value Description		Raw value of Parameter
RAW_VALUE_TYPE		Specifies type of raw value where the parameter is decalibrated to
RAW VALUE SIZE IN BITS		Size of the raw value in bits
RAW VALUE SIZE IN BYTES		Size of the raw value in bytes (For string parameter)
FLOAT_RAW_VALUE RANGE	MIN_FLOAT .. MAX_FLOAT	Range of Raw Value of type Float
INTEGER_RAW_VALUE RANGE	MIN_INTEGER .. MAX_INTEGER	Range of Raw Value of type Integer
UNSINGED_INT_RAW_VALUE RANGE	0 .. MAX_UNSIGNED	Range of Raw Value of type Unsigned_Integer
Engineering Value Description		Engineering value of Parameter
INTEGER_ENG_VALUE RANGE	MIN_INTEGER .. MAX_INTEGER	Range of Engineering Value of type Integer
FLOAT ENG RANGE	MIN_FLOAT .. MAX_FLOAT	Range of Engineering Value of type Float
ENGINEERING_UNITS		Engineering Unit for Parameter; optional;

Aggregates	Standard Value / Default (*)	Comment
Discrete Decalibration		Describes decalibration for discrete parameter
PARA_DISCRETE_CALIBRATION up to 31 Statecode / Raw Value Pairs		Used for decalibration of discrete parameter (type STATE_CODE) Statecodes are strings up to 8 characters The statecode ”OTHER” is reserved and must not be used.
Analog Decalibration		Describes decalibration for float/integer/unsigned_integer parameter
CURVE TYPE	POLYNOM	mandatory IDENTICAL,POLYNOM or POINT_PAIRS. For VICOS: IDENTICAL means same as POLYNOM with coefficients 0.0,1.0,0.0,0.0,0.0
ANALOG POINT PAIRS		not used
ANALOG_DECAL_COEFFICIENTS	0.0, 1.0, 0.0, 0.0, 0.0	mandatory (if curve type = polynom)
ANALOG DECAL POINT PAIRS	(MIN_FLOAT, MIN_FLOAT) (MAX_FLOAT, MAX_FLOAT)	(if curve type = point pairs)
Formal Parameter Definition		Formal Definition of the parameter
FORMAL PARAMETER SOURCES	()	Defines parameter that may be used during the UCL ISSUE command to replace a value within the data part. The place, where the actual parameter‘s value is replaced within the packet, is specified by the PARAMETER_LIST aggregate. FORMAL PARAMETER SOURCES defines in UCL syntax the definition of the parameter as well as its default value Up to 255 Parameter may be specified. Only ”IN” parameter are allowed. For parameter types refer to consistency checks below. Example of UCL Syntax: Default_Cmd_Id: integer := 1; (Cmd_Id: in INTEGER := Default_Cmd_Id ; Cmd_Value: String := ”Execute_Function”; Function: in string); Declarations of e.g. constants may be used as in any UCL program. Must be compiled/stored via the CLS Editor. Then the definition of the parameters is stored in other aggregates of the MDB.

Aggregates	Standard Value / Default (*)	Comment
PARAMETER LIST		For each formal parameter specified above, the location and size of the field where the actual parameter's value shall be replaced during runtime, is to be given.
Parameter_Name		Name of the parameter. This name may be used in the UCL ISSUE command to specify the parameter. It is defined in the UCL syntax above.
Parameter_Location		offset in bits to the beginning of the data field of the packet
Parame_Number_of_bits		number of bits allowed values: INTEGER_TYPE: 1..32 STRING_TYPE: 8..4096 REAL_TYPE: 32 TIME_TYPE: 5*8 = 40 (converted to unsegmented time code format) PATH_NAME_TYPE 32 (pathname is converted to SID in packet) UNSIGNED_INTEGER_TYPE 1..32 STATE_CODE_TYPE 1..32 (is decalibrated to unsigned integer)
Verification Description		
COMMAND_VERIFICATION TIMES		Times to wait before verification starts after sending See EGSE_ANALOG_STIMULUS
COMMAND_VERIFICATION		Defines the enditems to be checked and the values to be checked See EGSE_ANALOG_STIMULUS
Command Authorization		
CRITICAL_COMMAND	false (*)	true / false If "true", the command is to be authorized via password before sending
PASSWORD	empty	String up to 8 character Defines the password which has to be entered before sending

Consistency Checks:

(1) Mode of Formal Parameters must be: IN

(2) Type of Formal Parameters must be one of:

STRING_TYPE
STATE_CODE_TYPE
INTEGER_TYPE
REAL_TYPE
PATHNAME_TYPE
TIME_TYPE
UNSIGNED_INTEGER_TYPE

(3) Up to 255 parameter are allowed.

(4) For each formal parameter (in aggregate T_FORMAL_PARAMETERS), there must be an entry in the Parameter Insertion List (aggregate T_LIST_OF_PARAMETERS)

(5) Each Predefined Value shall be entirely located within the bounds of the data buffer, i.e.

LOCATION + VALUE_SIZE – 1 <= (Buffer Length) * 8

(6) Predefined Values shall not overlap one another, i.e.:
For any Predefined Value i (i = 1 .. N–1)

$LOCATION_{(i+1)} > LOCATION_{(i)} + VALUE\_SIZE_{(i)} - 1$

with

LOCATION :	T_GENERAL_BITSTREAM_LAYOUT. **Location**
VALUE_SIZE:	T_INTEGER_DEFINITION.**Number of Bits**, for integer
	64, for float
	(*size_of* T_BINARY_DEFINITION.**Value**) * 8 , for strings
Buffer Length:	T_DATA_BUFF_LAYOUT_GLOB_LENGTH.**Global Length**
N:	Number of entries (predefined values) in the data buffer

(7) Each Parameter Value shall be entirely located within the bounds of the data buffer, i.e.

PARAM_LOC + PARAM_SIZE – 1 <= (Buffer Length) * 8

with

PARAM_LOC :	T_LIST_OF_PARAMETERS.**Parameter Location**
PARAM_SIZE:	T_LIST_OF_PARAMETERS.**Parameter Number of Bits**
Buffer Length:	T_DATA_BUFF_LAYOUT_GLOB_LENGTH.**Global Length**

(8) For checks on COMMAND_VERIFICATION see EGSE_ANALOG_STIMULUS

7.6.3.6.4 EGSE_PREDEFINED_TC

This is a parameterizable MDB end–item defining a complete CCSDS packet with header and data information.

This enditem defines a CCSDS telecommand to be sent to the unit under test by SAS. Similar to the other stimulus definitions, it contains some general information, physical address information and a parameter list. In addition, however, the necessary data to describe the layout of the CCSDS packet has to be provided in the following properties:

- a *CCSDS (primary) header description* which is **mandatory**. This property defines the contents of all fields of the primary header of the CCSDs packet which have to be predefined before packet generation.
- a *CCSDS secondary header description* which is **optional**. This property defines the contents of all fields of the secondary header of the CCSDs packet which have to be predefined before packet generation.
- a *bitstream layout description* which is **mandatory**. This property describes the static bit pattern to be put into the packet data field (after the secondary header, if present or after the primary header if no secondary header available). The bitstream layout can be used to define op–codes for specific onboard commands, for example.
- a *command buffer item description* which is **optional**. This description is not needed for checkout but for simulation purposes only.
- a *Command Verification Description*, which is **optional**

  See EGSE_ANALOG_STIMULUS
- up to 255 *parameters*, and for each:
  - a raw value description
  - an engineering value description
  - a decalibration description which is optional. Depending on the parameter type, this may be analog (INTEGER, REAL) or discrete (STATE_CODE) decalibration

The meaning of the formal parameter list defined with this enditem is somehow different from the one defined for the other stimuli enditems. Here, the formal parameters defined for the telecommand will be substituted at run–time into the data section of the CCSDS packet (see above for a definition) by CGS. The logic with respect to the handling of the bitstream layout and parameters is that first the static bit patterns defined by bitstream layout is put into the data section and then the actual values of the formal parameters will be substituted, possibly overwriting the static bits.

The enditem is only used for checkout (VICOS).

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
STIMULUS GENERAL INFO (refer to EGSE_ANALOG_STIMULUS)		General Attributes
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical address eh-were to sent the stimulus/GDU to
Bitstream Layout		Definition of predefined bitstreams (data) for the binary packet
GENERAL_BITSTREAM_LAY-OUT Refer to EGSE_BINARY_PACKET		A list of bitstream items which shall be inserted into the packet as predefined values.
INTEGER_DEFINITION Refer to EGSE_BINARY_PACKET		
FLOAT_DEFINITION Refer to EGSE_BINARY_PACKET		
BINARY_DEFINITION Refer to EGSE_BINARY_PACKET		
Parameter Raw Value Description		Raw value of Parameter
RAW_VALUE_TYPE		Specifies type of raw value where the parameter is decalibrated to
RAW VALUE SIZE IN BITS		Size of the raw value in bits
RAW VALUE SIZE IN BYTES		Size of the raw value in bytes (For string parameter)
FLOAT_RAW_VALUE RANGE	MIN_FLOAT .. MAX_FLOAT	Range of Raw Value of type Float
INTEGER_RAW_VALUE RANGE	MIN_INTEGER .. MAX_INTEGER	Range of Raw Value of type Integer
UNSINGED_INT_RAW_VALUE RANGE	0 .. MAX_UNSIGNED	Range of Raw Value of type Un-signed_Integer
Engineering Value Description		Engineering value of Parameter
INTEGER_ENG_VALUE RANGE	MIN_INTEGER .. MAX_INTEGER	Range of Engineering Value of type Integer
FLOAT ENG RANGE	MIN_FLOAT .. MAX_FLOAT	Range of Engineering Value of type Float
ENGINEERING_UNITS		Engineering Unit for Parameter; optional;

Aggregates	Standard Value / Default (*)	Comment
Decalibration/Calibration		Describes decalibration for parameter
Discrete Decalibration		Describes decalibration for discrete parameter
PARA_DISCRETE_CALIBRATION up to 31 Statecode / Raw Value Pairs		Used for decalibration of discrete parameter (type STATE_CODE) Statecodes are strings up to 8 characters The statecode ”OTHER” is reserved and must not be used.
Analog Decalibration		Describes decalibration for float/integer/unsigned_integer parameter
CURVE TYPE	POLYNOM	mandatory IDENTICAL,POLYNOM or POINT_PAIRS For VICOS: IDENTICAL means same as POLYNOM with coefficients 0.0,1.0,0.0,0.0,0.0
ANALOG POINT PAIRS		not used
ANALOG_DECAL_COEFFICIENTS	0.0, 1.0, 0.0, 0.0, 0.0	mandatory (if curve type = polynom)
ANALOG DECAL POINT PAIRS		(if curve type = point pairs)
Formal Parameter Definition		Formal Definition of the parameter
FORMAL PARAMETER SOURCES	()	Defines in UCL syntax the definition of the parameter as well as its default value Up to 255 Parameter may be specified. For Syntax and restrictions refer to EGSE_BINARY_PACKET
PARAMETER LIST refer to EGSE_BINARY_PACKET		For each formal parameter specified above, the location and size of the field where the actual parameter’s value shall be replaced during runtime, is to be given.

Aggregates	Standard Value / Default (*)	Comment
CCSDS header		Header of the CCSDS Packet
CCSDS HEADER DESCRIPTION		Primary Header of CCSDS Packet
Version	0	Version of the CCSDS service
Type	0	0 = realtime, 1 = playback
Secondary Header	TRUE	Defines if a secondary is applicable. COF/ISSA packets have secondary header
Application ID	empty	Identifies receiver of packet Integer range 0..
Sequence Flags	3	Only value 3 (Unsegmented) is currently supported
Packet Length		size of total CCSDS packet (excl. header): <Packet Length in bytes> – 1 with <Packet Length in bytes> being the number of bytes from the first byte following the primary header to the last byte in the packet. 1..4095 (bytes); (The minimum length of CCSDS packets is 2 bytes, the maximum 4096 bytes)
CCSDS SECOND HEADER		Secondary Header of CCSDS Packet.
Checksum_indicator	TRUE	Indicates, if checksum is available at the end of the packet. Must always be TRUE
Time_Id	TIME_OF_PACKET_GENERATION	Allowed: TIME_OF_PACKET_GENERATION NO_TIME Note: The value of TIME ID is overwritten by the online system (TES) when the CCSDS packet is sent. Refer to description of the UCL ISSUE command in App.I
Packet_Type	SYSTEM_COMMAND	A value from a set of predefined packet types for COF/ISSA may be selected. Only the types SYSTEM_COMMAND , ESSENTIAL_COMMAND or PAYLOAD_COMMAND are allowed
Packet ID	If omitted, the SID of this end item will be used as Packet–ID.	Identifies the structure and the function of the particular packet.

Aggregates	Standard Value / Default (*)	Comment
Verification Description		
COMMAND_VERIFICATION TIMES		Times to wait before verification starts after sending See EGSE_ANALOG_STIMULUS
COMMAND_VERIFICATION		Defines the enditems to be checked and the values to be checked See EGSE_ANALOG_STIMULUS
Command Authorization		
CRITICAL_COMMAND	false (*)	true / false If “true”, the command is to be authorized via password before sending
PASSWORD	empty	String up to 8 character Defines the password which has to be entered before sending

The general format of CCSDS packet headers is shown below.

Primary Header							**User Data Field**
Packet Identification				**Sequence Control**			
Version	**Type**	**Secon–dary Hdr Flag**	**AP ID**	**Sequence Flags**	**Sequence Count**	**Packet Length**	
3 bits	**1 bit**	**1 bit**	**11 bits**	**2 bits**	**14 bits**	**16 bits**	
6 Octets							**Var**

CCSDS Primary Header	*User Data Field*							
	Secondary Header						*Data*	*Check sum*
	Time	*User Data Control Field*				*Packet ID*		
		Time ID	*Check sum Indi cator*	*Spare*	*Packet Type*			*[op-tional]*
	5 octets	*2 bits*	*1 bit*	*1 bit*	*4 bits*	*4 octets*		*16 bits*
6 octets	*10 octets*						*var*	

CCSDS Primary & Secondary Headers

Consistency Checks:

(1) Mode of Formal Parameters must be: IN

(2) Type of Formal Parameters must be one of:

STRING_TYPE
STATE_CODE_TYPE
INTEGER_TYPE
REAL_TYPE
PATHNAME_TYPE
TIME_TYPE
UNSIGNED_INTEGER_TYPE

(3) Up to 255 formal parameters are allowed.

(4) For each formal parameter (in aggregate T_FORMAL_PARAMETERS), there must be an entry in the Parameter Insertion List (aggregate T_LIST_OF_PARAMETERS)

(5) If the CCSDS Packet has a secondary header, then the range of the Packet Length Field shall be 9 .. 4095 (since the secondary header is 10 bytes long), i.e.

if T_CCSDS_HEADER_DESCRIPTION.**Secondary Header** = ’TRUE’ then
9 <= T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field** <= 4095

(6) Each Predefined Value shall be entirely located within the bounds of the data buffer, i.e.

LOCATION + VALUE_SIZE – 1 <= (Packet Length Field + 1) * 8

(7) Predefined Values shall not overlap one another, i.e.:
For any Predefined Value i (i = 1 .. N–1)

$\text{LOCATION}_{(i+1)} > \text{LOCATION}_{(i)} + \text{VALUE\_SIZE}_{(i)} - 1$

with

LOCATION :	T_GENERAL_BITSTREAM_LAYOUT. **Location**
VALUE_SIZE:	T_INTEGER_DEFINITION.**Number of Bits**, for integer
	64, for float
	(size_of T_BINARY_DEFINITION.**Value**) * 8 , for strings
Packet Length Field:	T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field**
N:	Number of entries (predefined values) in the data buffer

(8) Each Parameter Value shall be entirely located within the bounds of the data buffer, i.e.

PARAM_LOC + PARAM_SIZE – 1 <= (Packet Length Field + 1) * 8

with

PARAM_LOC :	T_LIST_OF_PARAMETERS.**Parameter Location**
PARAM_SIZE:	T_LIST_OF_PARAMETERS.**Parameter Number of Bits**
Packet Length Field:	T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field**

(9) For checks on COMMAND_VERIFICATION see EGSE_ANALOG_STIMULUS

7.6.3.6.5 INTEGER_STIMULUS

This enditem is used by CSS/CMAS only.

The Integer Stimulus type is a *parameterized* end–item type that takes one integer value as parameter.The formal definition is:

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical address
Engineering Value Description		Engineering value of Parameter
INTEGER ENGINEERING RANGE	MIN_INTEGER .. MAX_INTEGER	Range of Engineering Value
Raw Value Description		Raw value of Parameter
RAW_VALUE_TYPE	INTEGER	Specifies type of raw value where the parameter is calibrated to. allowed: INTEGER and UN-SIGNED_INTEGER
RAW VALUE SIZE IN BITS	32	Size of the raw value; 1..32;
INTEGER_RAW_VALUE_RANGE	MIN_INTEGER .. MAX_INTEGER	Range of Raw Value
UNSIGNED_INT_RAW_VALUE RANGE	MIN_UNSIGNED..MAX_UNSIGNED	Range of Raw Value
Decalibration/Calibration		Describes decalibration (for VI-COS) resp. calibration (for CSS)
CURVE TYPE	POINT_PAIRS	mandatory
ANALOG POINT PAIRS	(MIN_INTEGER, MIN_INTEGER) (MAX_INTEGER, MAX_INTEGER)	mandatory
ANALOG_DECAL_COEFFI-CIENTS		not used
ANALOG DECAL POINT PAIRS		not used
Formal Parameter Definition		Formal Definition of the parameter
FORMAL PARAMETER SOURCES	(PARAM: INTEGER)	Defines in UCL syntax the defini-tion of the parameter as well as its default value Must be compiled/stored via the CLS Editor. Then the definition of the parameter is stored in other ag-gregates of the MDB.

Consistency Checks:

(1) Only one parameter allowed.
(2) Mode of Formal Parameter must be: IN
(3) Type of Formal Parameter must be: INTEGER_TYPE

7.6.3.6.6 UNSIGNED_INTEGER_STIMULUS

The formal definition is:

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical address
Engineering Value Description		Engineering value of Parameter
UNSIGNED_INT_ENG RANGE	MIN_UNSIGNED..MAX_UNSIGNED	Range of Engineering Value
Raw Value Description		Raw value of Parameter
RAW_VALUE_TYPE	UNSIGNED_INTEGER	Specifies type of raw value where the parameter is calibrated to. allowed: UNSIGNED_INTEGER
RAW VALUE SIZE IN BITS	32	Size of the raw value; 1..32;
UNSIGNED_INT_RAW_VALUE RANGE	MIN_UNSIGNED..MAX_UNSIGNED	Range of Raw Value
Calibration		Describes calibration
CURVE TYPE	POINT_PAIRS	mandatory
ANALOG POINT PAIRS	(MIN_UNSIGNED,MIN_UNSIGNED) (MAX_UNSIGNED, MAX_UNSIGNED)	mandatory
ANALOG_DECAL_COEFFICIENTS		not used
ANALOG DECAL POINT PAIRS		not used
Formal Parameter Definition		Formal Definition of the parameter
FORMAL PARAMETER SOURCES	(PARAM: UNSIGNED_INTEGER)	Defines in UCL syntax the definition of the parameter as well as its default value Must be compiled/stored via the CLS Editor. Then the definition of the parameter is stored in other aggregates of the MDB.

Consistency Checks:

(1) Only one parameter allowed.
(2) Mode of Formal Parameter must be: IN
(3) Type of Formal Parameter must be: UNSIGNED_INTEGER_TYPE

7.6.3.6.7 DOUBLE_FLOAT_STIMULUS

The Double Float Stimulus type is a *parameterized* end–item type that takes one double–float value as parameter. It is used by CSS/CMAS only.

The formal definition is:

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical address
Engineering Value Description		Engineering value of Parameter
DOUBLE_FLOAT_ENG RANGE	MIN_DOUBLE_FLOAT..MAX_DOUBLE_FLOAT	Range of Engineering Value
Raw Value Description		Raw value of Parameter
RAW_VALUE_TYPE	UNSIGNED_INTEGER	Specifies type of raw value where the parameter is calibrated to. allowed: UNSIGNED_INTEGER or INTEGER
RAW VALUE SIZE IN BITS	32	Size of the raw value; 1..32;
UNSIGNED_INT_RAW_VALUE RANGE	MIN_UNSIGNED..MAX_UNSIGNED	Range of Raw Value
INTEGER_RAW_VALUE_RANGE	MIN_INTEGER .. MAX_INTEGER	Range of Raw Value
Calibration		Describes calibration
CURVE TYPE	POINT_PAIRS	mandatory
ANALOG POINT PAIRS	(0,0) (MAX_DOUBLE_FLOAT, MAX_DOUBLE_FLOAT)	mandatory
ANALOG_DECAL_COEFFICIENTS		not used
ANALOG DECAL POINT PAIRS		not used
Formal Parameter Definition		Formal Definition of the parameter
FORMAL PARAMETER SOURCES	(PARAM: LONG_REAL := 0.0)	Defines in UCL syntax the definition of the parameter as well as its default value Must be compiled/stored via the CLS Editor. Then the definition of the parameter is stored in other aggregates of the MDB.

Consistency Checks:

(1) Only one parameter allowed.
(2) Mode of Formal Parameter must be: IN
(3) Type of Formal Parameter must be: LONG_REAL_TYPE

7.6.3.6.8 BOOLEAN_STIMULUS

This enditem is used by CSS/CMAS only.

The Boolean Stimulus type is a *parameterized* end–item type that takes one boolean value as parameter. The formal definition is:

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical address
Raw Value Description		Raw value of Parameter
RAW_VALUE_TYPE	INTEGER	Specifies type of raw value Only INTEGER is supported
RAW VALUE SIZE IN BITS	1	Size of the raw value;
INTEGER_RAW_VALUE RANGE	MIN_INTEGER.. MAX_INTEGER	Range of Raw Value
Calibration		Describes calibration
BOOLEAN_CALIBRATION TRUE_Calibration FALSE_Calibration	 1 (*) 0 (*)	Defines calibration (raw values) for TRUE and FALSE Raw Value for TRUE Raw value for FALSE
Formal Parameter Definition		Formal Definition of the parameter
FORMAL PARAMETER SOURCES	(PARAM: BOOLEAN := FALSE)	Defines in UCL syntax the definition of the parameter as well as its default value Must be compiled/stored via the CLS Editor. Then the definition of the parameter is stored in other aggregates of the MDB.

Consistency Checks:

(1) Only one parameter allowed.
(2) Mode of Formal Parameter must be: IN
(3) Type of Formal Parameter must be: BOOLEAN_TYPE

7.6.3.6.9 PULSE_STIMULUS

The enditem is only used by CSS/CMAS;

The pulse stimulus is a stimulus which creates a pulse.
The formal definition is:

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical address
Raw Value Description		Raw value of Parameter
RAW_VALUE_TYPE	INTEGER	Specifies type of raw value Only INTEGER is supported
RAW VALUE SIZE IN BITS	1	Size of the raw value;
INTEGER_RAW_VALUE RANGE	MIN_INTEGER.. MAX_INTEGER	Range of Raw Value
Calibration		Describes calibration
BOOLEAN_CALIBRATION		Defines calibration (raw values) for TRUE and FALSE
TRUE_Calibration	1 (*)	Raw Value for TRUE
FALSE_Calibration	0 (*)	Raw value for FALSE

Consistency Checks:

7.6.3.6.10 BURST_PULSE_STIMULUS

The enditem is only used by CSS/CMAS;

The formal definition is:

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)		optional Describes the physical address
Engineering Value Description		Engineering value
UNSIGNED_INT_ENG RANGE	0..MAX_UNSIGNED	Range of Engineering Value
Raw Value Description		Raw value of Parameter
RAW_VALUE_TYPE	INTEGER	Specifies type of raw value Only INTEGER is supported
RAW VALUE SIZE IN BITS	1	Size of the raw value;
UNSIGNED_INT_RAW_VALUE RANGE	0.. MAX_UNSIGNED	Range of Raw Value
Calibration		Describes calibration
CURVE TYPE	POINT_PAIRS	mandatory
ANALOG POINT PAIRS	(0,0) (MAX_UNSIGNED, MAX_UNSIGNED)	mandatory
ANALOG_DECAL_COEFFI-CIENTS		not used
ANALOG DECAL POINT PAIRS		not used

Consistency Checks:

7.6.3.7 SWOP Commands and Response Packets

7.6.3.7.1 SWOP_COMMAND

This is a parameterizable MDB end–item defining a specific CCSDS packet for COF/ISSA.

This enditem defines a packet that is sent to the Onboard System (via a SAS) as a SWOP command.
A SWOP Command is defined by its SID and the set of parameters. A SWOP command is normally answered by the onboard system with a telemetry packet (response packet).

Similar to other stimulus definitions, it contains some general information and a parameter list. No physical address information is necessary, as the sending path is defined by the Application ID , which is given as an online parameter (derived from the CCSDS End Point). The CCSDS Primary Header is created by the sending software (VICOS/TES). The following information describes a SWOP command:

- a *CCSDS secondary header description* which is **optional**. This property defines the contents of all fields of the secondary header of the CCSDs packet which have to be predefined before packet generation.
- up to 255 *parameters*
- a reference to a response packet which is to be received after the command has been sent
- a reference to the SWRU which executes the SWOP within the onboard system

The formal parameters will be substituted at run–time into the data section of the CCSDS packet (see above for a definition) by VICOS/TES according to a predefined, fixed schema. No decalibration is foreseen.

The enditem is only used for checkout (VICOS) resp. by the onboard software.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
STIMULUS GENERAL INFO (refer to EGSE_ANALOG_STIMULUS)		General Attributes
RESPONSE PACKET_REFERENCE	empty (*)	Reference (Pathname) to an enditem of type RESPOINSE_PACKET describing the response packet of the SWOP
SWRU_REFERENCE	empty	Reference (Pathname) to an Software Replacable_Unit which executes the command onboard
CCSDS header		Header of the CCSDS Packet Only the secondary header is defined in the MDB. The primary header is completely derived within the online software.
CCSDS SECOND HEADER		Secondary Header of CCSDS Packet.
Checksum_indicator	TRUE	Indicates, if checksum is available at the end of the packet. Must always be TRUE
Time_Id	NO_TIME	Allowed: TIME_OF_PACKET_GENERATION: The actual Simulated Mission Time is set when sending the command. NO_TIME: No time is set when sending the command. Note: For handling of TIME ID in the online system (TES) when the CCSDS packet is sent refer to description of the UCL ISSUE_SW_COMMAND procedure in App.I
Packet_Type	SYSTEM_COMMAND	A value from a set of predefined packet types for COF/ISSA may be selected. Only the types SYSTEM_COMMAND , ESSENTIAL_COMMAND or PAYLOAD_COMMAND are allowed
Packet ID	If omitted, the SID of this end item will be used as Packet–ID. If an enditem is mapped to the type SWOP_COMMAND, the SID of this enditem is applicable	Identifies the structure and the function of the particular packet. Must conform to the definitions in the Onboard DMS.

Aggregates	Standard Value / Default (*)	Comment
Formal Parameter Definition		Formal Definition of the parameter
FORMAL PARAMETER SOURCES	()	Defines in UCL syntax the definition of the parameter as well as its default value Up to 255 Parameter may be specified. For Syntax and restrictions refer to EGSE_BINARY_PACKET The mapping of the parameters to fields in the ccsds packet sent up as a swop command and the response packets is as follows: 1. The ”in” parameter are inserted into the data part of the CCSDS packet in the sequence as they are defined. The length depends on the type of parameter: STRING_TYPE: No_of_Chars * 8 bits STATE_CODE_TYPE 8 bytes (64 bits) INTEGER_TYPE 32 bits REAL_TYPE 32 bits, IEEE Format PATHNAME_TYPE 32 bits (converted to SID) TIME_TYPE CCSDS unsegmented time code 2. The ”out” parameter are expected in the data part of the response packet following the Transaction_ID (4 bytes) and the Result (16 bits) with the encoding/length as shown for the ”in” parameters. Note: Transaction_ID and Result are fixed parameters of the ISSUE_SW_COMMAND procedure (UCL) of the GROUND_TO_ONBOARD System Library and thus must not be declared again as Formal Parameters of the SWOP Command.

Consistency Checks:

(1) Only Formal Parameters of mode IN or OUT are allowed.
(2) Type of Formal Parameters must be one of:

STRING_TYPE	(with fixed max_length)
STATE_CODE_TYPE	(with fixed state code list)
INTEGER_TYPE	(optionally with range constraint)
REAL_TYPE	(optionally with range constraint)
PATHNAME_TYPE	(optionally restricted to fixed set of item types)
TIME_TYPE	

(3) Checksum Indicator in aggregate T_CCSDS_SECOND_HEADER must be TRUE

(4) Packet Type in T_CCSDS_SECOND_HEADER must be ”System_Command” or ”Essential_Command” or ”Payload_Command”
(5) Time ID in T_CCSDS_SECOND_HEADER must be NO_TIME_FIELD or TIME_OF_PACKET_GENERATION

(6) The maximum number of parameters is 255.

7.6.3.7.2 RESPONSE_PACKET

Defines a CCSDS packet issued in response to a SWOP Command.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
RESPONSE PACKET		Describes the response packet of a SWOP
Private_ID		A private identifier for the response packet (cf. ADU Private ID
Sas_Reference		must contain pathname for SAS providing the ADU, which contains the CCSDS packet as the response
Checksum Indicator	TRUE	Indicates whether the response packet has a checksum or not
Packet ID	This attribute is optional. If omitted, the SID of this enditem will be used as default Packet ID.	Identifies a particular Response Packet.
		Note: For definition of return parameters contained in the Response Packet's data part refer to the definition of Formal Parameters of a SWOP_Command: Return Parameter must be defined as OUT – Parameter of the SWOP_Command.

7.6.3.7.3 APPLICATION_ID (APID)

In COF/ISSA, the logical path between two addresses is defined by one application ID (APID). A pair of socalled CCSDS Endpoints is allocated to an APID. When sending data from one end point to another end-point, the APID table defines the Application ID to be used.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
APID_TABLE		Allocation of a pair of CCSDS End-points to an Application ID. Also a device address is specified to allow for physical addressing of a SWOP Command.
Application ID (APID)		The application id assigned to the source and to destination direction of the CCSDS link.INTEGER range 0 .. 2047
Type	0	This attribute defines whether the Application Id is for a System or Payload Packet Value 0 : Application ID defines a System Packet Value 1 : Application ID defines a Payload Packet
Source_CCSDS End Point		References first end item of type CCSDS_End_Point (Pathname)
Destination_CCSDS_End Point		References second end item of type CCSDS_End_Point (Pathname)
Device Address	”UNDEFINED” (*)	Device Address to be used for the field of th e physical address with same name ,when a SWOP command is sent using this application ID pair.

Consistency Checks:

(1)The combination of the two attributes **Source CCSDS End Point** and **Destination CCSDS End Point** must be unique within the configuration scope.

(2) The combination of the two attributes **Application ID** and **Type** must be unique within the configuration scope.

7.6.3.7.4 CCSDS END POINT

In COF/ISSA, a logical path between two addresses on the CCSDS network is defined. Each address is called a CCSDS Endpoint. In the DB, each Endpoint is defined as an enditem, referencing the SWRU which is implementing the end point within the onboard system.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
SWRU_REFERENCE		Reference (Pathname) to an Software Replacable_Unit

Consistency Checks:

7.6.3.8 ADUs

ADUs are the generic data packets received by CGS from SAS. They can contain different type of data:
– list of individual raw values
– a binary data block
– telemetry packets (CCSDS packets).

CGS/VICOS unpacks ADUs received from SAS and extracts raw values of measurements from them.

Depending on the type of data to be sent by SAS to CGS, different MDB enditem types describe the layout and data contents of the ADUs. These are:

a. Structured ADU description. If the ADU sent by SAS contains a list of raw values of measurements, then a structured ADU description has to used in MDB to describe the layout of the ADU.
In this case, the SAS will have to write individual raw values into the ADU. The ADU description tells the SAS and CGS how and where to put/get the raw values. This type of an ADU is helpful in case the SAS receives individual raw values from dedicated front–end equipment (such as voltmeters, data acquisition units or switching matrices) or computes certain raw values itself.

b. Unstructured ADU description. If the ADU sent by SAS contains a binary data block (except CCSDS packets, which are a special case, see below), then an unstructured ADU description has to be used in MDB to describe the layout of the ADU.
In this case, the SAS will have to write a binary data block into the ADU(e.g. a data buffer read from a front end equipment. The ADU description tells CGS how and where to put/get the raw values. This type of an ADU is useful in case the SAS receives blocks of data from dedicated front–end equipment (such as parallel IO boards, Mil–Bus interfaces or DMA type equipment) and the SAS shall not interpret the data, but has to simply forward it to CGS.

c. The CCSDS packet ADU description. If the data block received by a SAS is a CCSDS telemetry packet, i.e. a data block with a given internal structure, then this type of ADU applicable. The SAS has to put the right CCSDS packet (depending on CCSDS application id, packet type and packet id) into the ADU and CGS will extract the raw values of the measurements from this packet then.

The information in MDB describing an ADU (i.e. the ADU Description) is passed to SAS when CGS needs the data contained in it or if the user explicitly requests acquisition of these data.

The overall properties of ADU Descriptions comprise:

- a *general description* which is mandatory**.** This description contains some identification information which is needed for CGS and the SAS to properly handle the data packet
- a *physical address*, which is optiona**l.** This property contains information needed by the SAS to physically acquire the data packet defined in the ADU and is not processed by CGS.
- a *measurement list* or *data buffer layout description* which is mandatory. This description tells CGS which measurements to extract from the ADU. In case the ADU is unstructured or CCSDS TM, additional information is provided in this aggregate to control where and how raw data have to be unpacked and formatted.
- a *description of the CCSDS (primary) header* which is optional**.** It is only needed for CCSDS packet ADUs and tells the SAS which CCSDS packet has to be put into the ADU.
- a *description of the CCSDS secondary header* which is optional**.** It is only needed of CCSDS packet ADUs and tells the SAS which CCSDS packet has to be put into the ADU.

7.6.3.8.1 STRUCTURED_ADU_DESCRIPTION

This enditem describes the layout of a structured ADU.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
ADU GENERAL INFO		General Info about the ADU
Private_Id	empty (*)	An ID which can be allocated to the ADU for SAS internal purposes.
Acquisition_Rate	1000	For ADUs that are acquired cyclically, the time in milliseconds between two acquisitions.
Sas_reference		Reference to the SAS (pathname) that delivers the ADU
Global_physical_address_required		The SAS may acquire all values of the ADU from this physical address
All_measurments_with_physical address		Or the SAS may acquire each measurement from a different physical address. Then each measurement carries such an address, and the SAS is required to use it. Note that specifying the Physical Address for each referenced measurement will significantly increase the amount of data stored in the SAS and transmitted over the network.
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)	empty (*)	Describes the global physical address of the ADU Is mandatory, if Global_physical_address_required is true
MEASUREMENT_ENDITEM_LIST		A list of references (pathnames) to measurements of type EGSE_..._MEASUREMENT The minimal implementation limit (in TES) of the number of referenced measurements in a Structured ADU Description is 68. It exactly depends on the contents of the measurements, in particular if the measurements are carrying physical address information. The overall maximum number is 100 references.

Consistency Checks:

7.6.3.8.2 UNSTRUCTURED_ADU_DESCRIPTION

This enditem describes the layout of an unstructured ADU.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
ADU GENERAL INFO		General Info about the ADU
Private_Id	empty (*)	An ID which can be allocated to the ADU for SAS internal purposes.
Acquisition_Rate		For ADUs that are acquired cyclically, the time in milliseconds between two acquisitions.
Sas_reference		Reference to the SAS (pathname) that delivers the ADU
Global_physical_address_required	TRUE (*)	The SAS acquires the whole packet from one physical address
All_measurments_with_physical address	FALSE (*)	N/A
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)	empty (*)	Describes the global physical address of the ADU
DATA_BUF_LAYOUT_END_ITEMS		List of measurements in the data part. A maximum number of 750 enditems may be given
Enditem		Reference to a measurement (enditem of type EGSE_..._MEASUREMENT)
Location		Location (in bits) of the first bit of this end item in the binary buffer. Note: No size is given here. The raw value size for each measurement is taken instead (refer to measurements: RAW VALUE SIZE IN BITS)
Data Source	empty (=DATA) (*)	Indicates Source of Buffer where Layout is defined for. Only applicable for CCSDS Packets (CCSDS_ADU_DE-SCRIPTION). For UNSTRUC-TURED ADU types it will be ig-nored (data part is always se-lected)
DATA_BUF_LAYOUT_GLOB LENGTH		Overall length of the data part of the ADU. Range 1..4096

Consistency Checks:

(1) Any measurement value to be contained in the ADU shall be entirely located within the bounds of the data buffer, i.e., the following condition shall be satisfied:

LOCATION + VALUE_SIZE – 1 <= (Buffer Length) * 8

(2) Measurement Values shall not overlap one another, that is:
For any measurement i (i = 1 .. N–1)

$$\mathrm{LOCATION}_{(i+1)} > \mathrm{LOCATION}_{(i)} + \mathrm{VALUE\_SIZE}_{(i)} - 1$$

with

LOCATION :	T_DATA_BUF_LAYOUT_END_ITEMS. **Location**
VALUE_SIZE:	T_RAW_VALUE_SIZE_IN_BITS.**Raw Value Size in Bits**
	or, for Byte Stream Measurements:
	T_RAW_VALUE_SIZE_IN_BYTES.**Raw Value Size in Bytes** * 8
Buffer Length:	T_DATA_BUFF_LAYOUT_GLOB_LENGTH.**Global_length**
N:	Number of entries (measurement values) in the data buffer

7.6.3.8.3 CCSDS_ADU_DESCRIPTION

This enditem describes the layout of a CCSDS TM packet ADU.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
ADU GENERAL INFO		General Info about the ADU
Private_Id	empty (*)	An ID which can be allocated to the ADU for SAS internal purposes.
Acquisition_Rate		N/A (it is expected, that CCSDS packets are sent asynchronously) Value can be specified,however, if SAS requires it
Sas_reference	TRUE (*)	Reference to the SAS (pathname) that delivers the ADU
Global_physical_address_required	FALSE (*)	The SAS acquires the whole packet from one physical address
All_measurments_with_physical address		N/A
EGSE PHYSICAL ADDRESS (refer to EGSE_INTEGER_MEASUREMENT)	empty (*)	Describes the global physical address of the ADU / CCSDS Packet
CCSDS header		Header of the CCSDS Packet
CCSDS HEADER DESCRIPTION		Primary Header of CCSDS Packet
Version	0	Version of the CCSDS service
Type	0	0 = realtime, 1 = playback
Secondary Header	TRUE	Defines if a secondary is applicable. COF/ISSA packets have secondary header
Application ID	empty	Identifies sender of packet Integer range 0..
Sequence Flags	3	Only value 3 (Unsegmented) is currently supported
Packet Length		size of total CCSDS packet (excl. header): <Packet Length in bytes> – 1 with <Packet Length in bytes> being the number of bytes from the first byte following the primary header to the last byte in the packet. 1..4095 (bytes); (The minimum length of CCSDS packets is 2 bytes, the maximum 4096 bytes)

CCSDS SECOND HEADER Checksum_indicator Time_Id Packet_Type Packet ID	 TRUE NO_TIME (*) DEFAULT_PACKET If omitted, the SID of this end item (i.e. the SID of the CCSDS_ADU_DESCRIPTION) will be used as Packet–ID. Note: If an enditem is mapped to the CCSDS_ADU_DESCRIP-TION, the SID of the mapped enditem is taken by default)	Secondary Header of CCSDS Packet. Indicates, if checksum is available at the end of the packet. Defines interpretation of the time A value from a set of predefined packet types for COF/ISSA may be selected. Type is not checked by CGS/VICOS when the packet is received. Identifies the structure and the function of the particular packet. For COF/ISSA, the packet id must match the SID used for the corresponding TM descriptor.
DATA_BUF_LAYOUT_END_ITEMS Enditem Location Data Source	 empty (=DATA) (*)	List of measurements in the data partof the CCSDS packet. A maximum number of 750 enditems may be given Reference to a measurement (enditem of type EGSE_..._MEASUREMENT) Location (in bits) of the first bit of this end item in the data part of the CCSDS Packet. (i.e. following the secondary header if present) (range 1..8*Packet_Length) Note: No size is given here. The raw value size for each measurement is taken instead (refer to measurements: RAW VALUE SIZE IN BITS) Indicates Source of Buffer where Layout is defined for: Data Part , Primary Header Part , Secondary Header Part or Checksum If not specified, the DATA part of the packet is applicable.

Consistency Checks:

(1)The minimum length of CCSDS packets is 1 bytes;

the maximum length supported by CGS is 4090 (+ 6 bytes Primary Header).

The <Packet Length in bytes> is therefore an integer between 1 and 4090;

hence, the Packet Length Field must be in the range 0 to 4089.

(2) Any measurement value to be contained in the ADU shall be entirely located within the bounds of the data buffer, i.e., the following condition shall be satisfied:

LOCATION + VALUE_SIZE – 1 <= (Packet Length Field + 1) * 8
in case the DATA_SOURCE is set to DATA and no Secondary Header is defined

LOCATION + VALUE_SIZE – 1 <= (Packet Length Field –9) * 8
in case the DATA_SOURCE is set to DATA and a Secondary Header is defined

LOCATION + VALUE_SIZE – 1 <= 6 * 8
in case the DATA_SOURCE is set to HEADER

LOCATION + VALUE_SIZE – 1 <= 10 * 8
in case the DATA_SOURCE is set to SECONDARY_HEADER

(3) Measurement Values shall not overlap one another, that is:
For any measurement i (i = 1 .. N–1)

$$\mathrm{LOCATION}_{(i+1)} > \mathrm{LOCATION}_{(i)} + \mathrm{VALUE\_SIZE}_{(i)} - 1$$

(4) If Source is ”HEADER”, the Location must not exceed the value 48.
If Source is ”SECONDARY_HEADER”, the Location must not exceed the value 80.

(5) If the CCSDS Packet has no secondary header, then the DATA_SOURCE must not be set to SECONDARY_HEADER.

(6) If for any measurement to be contained in the ADU the DATA_SOURCE is set to HEADER or SECONDARY_HEADER, the RAW_VALUE_TYPE of the measurement must be one of the following alternatives:

UNSIGNED_INTEGER
SIGNED_INTEGER

with

LOCATION :	T_DATA_BUF_LAYOUT_END_ITEMS. **Location**
VALUE_SIZE:	T_RAW_VALUE_SIZE_IN_BITS.**Raw Value Size in Bits** or, for Byte Stream Measurements: T_RAW_VALUE_SIZE_IN_BYTES.**Raw Value Size in Bytes** * 8
Packet Length Field:	T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field**
N:	Number of entries (measurement values) in the data buffer

7.6.3.9 Simulated Data

In order to support the simulation mode on a test node, it is possible to define simulated data in the MDB for ADUs and measurements in a dedicated enditem type. These predefined data will be used as initial values when running in simulation mode.

7.6.3.9.1 SIMULATED_ADU_DESCRIPTION

This is an end–item describing the contents of an ADU (and their related measurements) in terms of simulated data.

The formal definition is:

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
REFERENCED_ADU		Pathname to that ADU which contents shall be simulated.
SIMULATED DATA GLOBAL LENGTH		Global_Length of ADU Can be lower than the length of the simulated ADU. Applicable for ADU of type UN-STRUCTURED or CCSDS
SIMULATED DATA LIST		For structured ADU only: list of values that shall be simulated for each enditem referenced in the ADU Order of list must be same as in ADU The values can be of different types: INTEGER, UNSIGNED_INTEGER, FLOAT, BYTESTREAM. The bytestream value is to be entered as a string, and not as hexadecimal values

Aggregates	Standard Value / Default (*)	Comment
SIMULATED RAW VALUE LIST Location Length:		Only if ADU is UNSTRUCTURED or CCSDS: List of values to be simulated for each enditem referenced in the ADU. Will be inserted into the the binary buffer of the UNSTRUCTURED_ADU resp. into the data field of the CCSDS_ADU Location of first bit of the value in the ADU data buffer resp. in the CCSDS data part Number of bits of the value **Attention:** <u>The list of values must match exactly the list of enditems in the referenced ADU with their bit location and type of raw value. Also the sequence of the items must be the same in both lists (e.g.: the n–th item of the raw value list must define an unsigned integer value of b bits at location l, if the n–th measurement of the referenced ADU has a raw value type unsigned integer and starts at location l with a raw value size of b bits.</u> <u>Only if the value is given as an HEXA value (Raw Value in Hex option is TRUE), this rule may be violated.</u>

Consistency Checks:

(1) Actual length of Simulated ADU must be less than or equal to the length of the referenced ADU,
i.e. if referenced ADU is a CCSDS ADU then:
Sim–ADU → T_SIMULATED_DATA_GLOBAL_LENGTH.**Global Length** <=
Ref–ADU → T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field** + 1

or if referenced ADU is an unstructured ADU then:
Sim–ADU → T_SIMULATED_DATA_GLOBAL_LENGTH.**Global Length** <=
Ref–ADU → T_DATA_BUFF_LAYOUT_GLOB_LENGTH.**Global Length**

(2) Any simulated measurement value in an Unstructured or CCSDS Simulated ADU shall be entirely located within the bounds of the data buffer, i.e., the following condition shall be satisfied:
(a) LOCATION + VALUE_SIZE – 1 <= (Buffer Length) * 8

(b) Further, measurement Values shall not overlap one another, that is:
For any measurement i (i = 1 .. N–1)
$\text{LOCATION}_{(i+1)} > \text{LOCATION}_{(i)} + \text{VALUE\_SIZE}_{(i)} - 1$

with

LOCATION :	T_SIMULATED_RAW_VALUE_LIST. **Location**
VALUE_SIZE:	T_SIMULATED_RAW_VALUE_LIST.**Length**
ADU Length:	T_SIMULATED_DATA_GLOBAL_LENGTH.**Global Length**
N:	Number of entries (measurement values) in the data buffer

(3) For any given Structured Simulated ADU (i.e. when type of referenced ADU = ’STRUCTURED’), the number of entries in the Simulated Data List (aggregate T_SIMULATED DATA_LIST) must not exceed the number of measurement end items in the Measurement End Item List (aggregate T_MEASUREMENT_END_ITEM_LIST) of the referenced ADU.

(4) The types of values specified in a Simulated ADU must be compatible with the types of the measurements at corresponding locations in the referenced ADU, i.e., the specified simulated data values (in aggregates T_SIMULATED_DATA_LIST or T_SIMLATED_RAW_VALUE LIST) shall be:

either **Integer Raw Value** or **Unsigned Integer Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_INTEGER_MEASUREMENT

Unsigned Integer Raw Value if the corresponding measurement in the referenced ADU is of type EGSE_DISCRETE_MEASUREMENT

Float Raw Value if the corresponding measurement in the referenced ADU is of type EGSE_FLOAT_MEASUREMENT

Byte Stream Raw Value if the corresponding measurement in the referenced ADU is of type EGSE_BYTESTREAM_MEASUREMENT

7.6.3.10 Lists

It is often useful to group certain types of enditems in lists. For this purpose, MDB provides two predefined lists, being:

– Monitor lists (Measurement Lists)
– Stimuli lists (GDU Description Lists)

Monitor lists can be used to collectively control groups of measurements, e.g. to enable them for monitoring, start_acquisition or enable processing Thus they can only contain references to other MDB items of type measurement, SW variable or Derived Values.

Stimuli lists can be used to define a predefined sequence of stimuli if this sequence is often used operationally. Thus stimuli lists can only contain references to other MDB items of type stimuli. The list has two parameters, FIRST and LAST, which at runtime or by MDB default values control which subsection of the list shall be used (e.g. a stimulus list contains references 1..100 to other stimuli and it is invoked with actual parameters FIRST=10 and LAST=20, then only elements 10..20 will be processed).

The overall aggregate structure of the list enditems is similar. They contain one aggregate which defines the list and only the stimuli lists contain another aggregate which defines the formal parameter list.

7.6.3.10.1 EGSE_MONITOR_LIST

This enditem defines a list of measurements and/or SW variables/Derived Values.

The structure of this enditem is rather simple and it contains only one aggregate. The formal definition is:

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
VALUE ENDITEM LIST		List of pathname references to measurements or SW_variables. A maximum of 500 entries can be defined.

7.6.3.10.2 GDU_DESCRIPTION_LIST

This enditem defines a list of stimuli or telecommands. Such a list can be executed in the online system giving a single ISSUE statement.

Item Properties:

Aggregates	Standard Value / Default (*)	Comment
STIMULI_END_ITEM_LIST		List of pathname references to stimuli or telecommands (i.e. enditems of type EGSE_..._STIMULUS, EGSE_BINARY_PACKET or EGSE_PREDEFINED_TC. A maximum of 500 entries can be defined.
FORMAL PARAMETER SOURCES	(FIRST: UNSIGNED_INTEGER := 1; LAST: UNSIGNED_INTEGER := <number of entries in the list>)	Defines in UCL syntax the definition of the parameter as well as its default value. Names FIRST and LAST are mandatory. Must be compiled/stored via the CLS Editor. Then the definition of the parameter is stored in other aggregates of the MDB.

Consistency Checks:

- Only 2 parameters allowed. They must be defined as follows:
- Parameter 1:
- NAME = First
  MODE = IN
  TYPE = UNSIGNED INTEGER
  Default value: = 1
- Parameter 2:
- NAME = Last
  MODE = IN
  TYPE = UNSIGNED INTEGER
  Default value: = number of entries in the list

7.7 Consistency Checking

7.7.1 Input Checking

Some items (e.g. value ranges) are checked already when entering the data in the DB input masks. In case the data entered is not valid, a beep together with an error message is generated and the data is not accepted.

7.7.2 Item Checking

For some items. a tool Check_MDB_Item may be called via the flexible tool interface (”Tools” Menu for the enditem).

The Check_MDB_Item program is implemented for the following enditem types:

MEAS	EGSE_xxx_MEASUREMENT EGSE_xxx_SW_VARIABLE EGSE_xxx_DERIVED_VALUE
ADU	xxx_ADU_DESCRIPTION
GDU	EGSE_PREDEFINED_TC EGSE_BINARY_PACKET EGSE_ANALOG_STIMULUS EGSE_DISCRETE_STIMULUS
SYNOPT	WDU_GROUND_SYNOPTIC_DISPLAY

The program performs predefined checks and displays the result to the user by opening a text window. In this window the errors found, a statistic on errors found and the enditem data loaded from the MDB is displayed.

In case there are errors reported by the loading software, a specific error log file is opened showing the error messages.

The program also searches for references in other enditems of the scope to the item under check. These are currently
– references of measurements/sw_variables/derived values to other measurements/sw_variables/ derived values via conditions
– references of synoptics to measurements/sw_variables/derived values
– references of ADU_DESCRIPTIONs to measurements

Note: The searching for references may take some time in large databases. Therefore the searching may be suppressed via the environment variable MDA_CHECK_MDB_ITEM (value: ”NO_REFERENCES”). Then all checks are performed, but the references from other items are not listed.

For alist of checks performed and their classification see Appendix N.

7.7.3 Consistency Checking on CDU/CCU Level

The consistency of a CDU or a CCU is checked via the ”Consistency Checker (CC)” Program. It may be called via the ”Command” menu of a CDU resp. the ”Command” menu of a CCU Version.

The CC defines the consistency status of a CDU/CCU.

For description of the CC see MDA Reference Manual [2.1.1.3]

7.7.4 Checking when Loading to Files

When creating the SCOE load files, the MDB data are checked within the scope of the loaded data again. Errors are reported to the user. (Refer to ch. 7.1.11).

8 INTEGRATION AND TEST

8.1 General Operation in the Checkout Environment

8.1.1 Checkout Operations

During checkout operations, CGS allows to monitor and command the Unit under Test and to simulate the environmet of the UUT or the EGSE. The DB servers, testnodes and workstations are active, the data is acquired via SAS and Frontends and the Simulation Model might be activated to simulate data via frontends to the UUT. Commands are given to the UUT/Onboard System and to the EGSE, reconfiguration requests are performed and acquired data stored in the TRDB during the test can be evaluated in parallel to ongoing test operations or offline after the test.

8.1.2 Operational Modes

CGS allows to operate for checkout in different modes:

- **EGSE NORMAL** mode
  EGSE is online and acquiring data from the UUT. Commands are sent to the UUT
  Test Nodes are operating in mode NORMAL
  SAS are connected to front ends. Front ends are active.
  SAS and Automated Procedures may run to access online data.
  Simulation on Simulation Nodes might be active or not.
  Simulation can be controlled via a specific window (MOCS) or via the HCI/HLCL Command Window (”Central Commanding”)
  TEV may be used to evaluate data online
- **EGSE Simulation** mode
  EGSE is online and acquiring data from the UUT. Commands are sent to the UUT
  Test Nodes are operating in mode SIMULATION. Data from frontends is internally simulated by test nodes. Commands are not sent.
  No SAS connecting to front ends are running . Front ends are inactive.
  SAS and Automated Procedures may run to access online data.
  Simulation on Simulation Nodes is not active.
  TEV may be used to evaluate data online
- **EGSE ”Mixed”** mode
  EGSE is online and acquiring data from the UUT. Commands are sent to the UUT
  Test Nodes are operating in mode NORMAL or SIMULATION.
  On test nodes in NORMAL mode: SAS are connected to front ends. Front ends are active.
  On test nodes in SIMULATION mode: No SAS connecting to front ends are running . Front ends are inactive.Data from frontends is internally simulated by test nodes. Commands are not sent.
  SAS and Automated Procedures may run to access online data.
  Simulation on Simulation Nodes might be active or not.
  Simulation can be controlled via a specific window (MOCS) or via the HCI/HLCL Command Window (”Central Commanding”)
  TEV may be used to evaluate data online

Another EGSE ”Mixed” Mode is the operation of test nodes in REPLAY in parallel to test nodes in SIMULATION.

- **EGSE Replay** mode
  EGSE is online and acquiring data from the UUT. Commands are sent to the UUT
  Test Nodes are operating in mode REPLAY
  No SAS are connected to front ends. Front ends are inactive.
  SAS and Automated Procedures may run to access online data.
  Simulation on Simulation Nodes is not active.
- **Standalone Simulation** mode
  Simulation on Simulation Nodes is active.
  Simulation is controlled via a specific window (MOCS)
  EGSE is not online .
  Test Nodes are not operating. HCI on workstations are not operating.
  No SAS are connected to front ends. Front ends are inactive.
  No SAS and Automated Procedures are running to access online data.
  TEV may be used to evaluate data offline after the simulation run.
- **Offline Evaluation** mode
  TEV is used to evaluate data offline.
  Simulation on Simulation Nodes is not active.
  EGSE is not online .
  Test Nodes are not operating.
  No SAS are connected to front ends. Front ends are inactive.
  No SAS and Automated Procedures are running to access online data.

8.1.3 Operational Configurations

For the different modes, CGS can be configured in different operational configurations.
An operational configuration is determined by

– The **CGS System Services** runnning

The CGS System Services consist of
– The Time Services
– The Services to startup processes on local or remote nodes
– The TRDB Processes (DBS)

– The **CGS Applications** running

The CGS Applications consist of
– The processes HCI, TES,TEV,TSCV,CSS (group of processes)
– other interactive tools (I_MDB, DDED, CLS Editor,FWDU,GWDU etc.)

– The **Test Configuration** loaded

A Test Configuration determines the workstations, test nodes and simulation nodes involved, the mode of the nodes and the data to be loaded.
A test configuration is defined in the MDB, loaded and executed by the TSCV tool.
As it is part of a specific CCU of the MDB, activation of a test configuration includes selection of a CCU for a test.

The minimum setup for the operational modes are as follows:

For all: Oracle Services and the CGS daemon process need to be running. They are started at boot time automatically.

Dependent on the operational modes, additional system services and applications are requried. They are setup using the TSCV program or (for system services) the task selector's menu.

- **EGSE NORMAL, Simulation** mode
  – All System Services are running
  – Applications on Test Nodes (TES) and Workstations (HCI) need to be runing
  – The Message Window is required
  – TSCV is required to setup the configuration
  – Interactive Tools (incl. TEV) may be used at any workstation in parallel
- **EGSE Simulation** mode
  – All System Services are running
  – Applications on Test Nodes (TES) and Workstations (HCI) need to be runing
  – The Message Window is required
  – TSCV is required to setup the configuration
  – Interactive Tools (incl. TEV) may be used at any workstations in parallel
- **EGSE Replay** mode
  – All System Services are running
  – Applications on Test Nodes (TES) and Workstations (HCI) need to be runing
  – The Message Window is required
  – TSCV is required to setup the configuration
  – Interactive Tools (incl. TEV) may be used at any workstation in parallel
- **Standalone Simulation** mode
  – All System Services are running
  – Applications on Simulation Nodes (CSS Group) need to be runing
  – Workstations: the MOCS windows allow to control the simulation, no HCI is necessary.
  – The Message Window is required
  – Interactive Tools (incl. TEV) may be used at any workstation in parallel
- **Offline Evaluation** mode
  – Only the DBS Processes must be setup (Can be done via task selector)
  – TEV is running on workstation(s)
  – The Message Window is required
  – Other Interactive Tools may be used at any workstations in parallel

8.1.4 Operational Constraints

In the following, a list of constraints is given, to describe what must be considered in the setup or during the checkout operation.

Test Configurations

- A Test Configuration (TC) is loaded by TSCV into one of 5 predefined slots.
- Each TC has a CCU associated. By definition, this is the CCU where the TC is loaded from.
- TC define the role of each node for a test: It defines, which nodes are participating, and which test node is the Master Test Processor (MTP).
- Each Test Configuration has a test session associated, where all results generated within the test configuration are collected. If no specific name is given, the ”Default Test Session” is selected.
- Up to 5 Test Configurations may be setup in parallel, each having different test nodes and workstations defined, but all having the same DB Server node and operating on the same TRDB. The result must go to different (named) test sessions, except one, which can go to the ”Default Test Session”.
- Test Sessions can be specified before the Test Configuration is set to active. Then the test session is created/opened during the setup.
  Test Sessions may be created and closed, however, at any time when a test configuration is active.
  A test session is always allocated to one specific test configuration, except for the Default Test Session, which is open always, also when no test configuration is active.
- Each test session may be setup with Final Archiving active or not. Final Archiving active ensures, that data is written to the Optical Disk during test operations, whenever a predefined threshold for disk usage is reached. The threashold is configurable in the DBS Configuration files.

  Two Optical Disk drives are supported for final archiving. I.e. a maximum of two test sessions may have Final Archiving active in parallel.

  When Final Archiving is enabled, the device media should be ready and mounted on the drives before the test session is created.
- A test configuration may be modified online in TSCV.
- Test Evaluation (via TEV) and Standalone Simulation can be performed without having a Test Configuration setup.

EGSE Normal Mode

- Before setting up the Test Configuration, it has to be ensured, that all frontends are booted up and in their correct state to be able to connect to CGS test nodes.

EGSE ”Mixed” Mode

- A mixed configuration of test nodes in different modes might be used to verify a part of the UUT/EGSE and simulating another part. There is no constraint on the number of nodes being in either mode.

EGSE Replay Mode

- Test Sessions replayed must be selected via TSCV. They must exist in the TRDB and the data must be online (i.e. not exported/archived to optical disk).
  The used Test Configuration must exist in the MDB.
- Before a test session is started for replay, it must be ensured, that the test node's local disks have enough capacity to take all archive files needed for the selected time frame.
  It may take a considerable amount of time to distribute big archive files to the local disk of the test nodes, before the test session can be started for replay.
- In replay mode, the synchronisation of all test nodes with respect to the data replayed is not fully guaranteed. Test Nodes are setup in the sequence they appear in the Test Configuration. The start commands are given to the test nodes with a delay of roughly 100 ms between them.
- In replay mode, when replaying data, it can happen, that messages generated from replay data are received by the message window in a slightly different order, especially when a high load is selected by increasing the replay speed factor (see below)

8.2 Setting-up the Test Environment

8.2.1 Introduction

This section consists of 3 parts. The first part provides a detailed user overview of the Test System Configuration and Verification (TSCV) software which is the primary tool used in setting–up the test environment. The second part shows by example how one can configure the generic test system to user specific test environment and start a test execution session using TSCV. The third part explains how to setup automatic archiving of an execution session to the Final Archive medium.

8.2.2 Test System Configuration and Verification (TSCV)

The role of TSCV within CGS is to prepare the EGSE for a test session, implying

- select and load Test Configuration templates (up to five) from the database
- initiate set–up with one of the loaded test configurations as template
- ensure that the DBS processes are up and running
- Set the global default CCU in MDB for an active test configuration
- ensure that the TES processes on the testnodes are up and running
- command the testnodes' test execution software (TES) to initialize and load configuration data from the Mission Database according to the Test Configuration template
- Notify HCI
- control the time services system (TSS) initialization
- command the testnodes with respect to their overall state (execution mode)
- initiate (open) and close test session in the Test Result DB
- distribute archived test result files to the testnodes for the purpose of test replays
- initialize and maintain the Test Result DB
- monitor the status of the test nodes
- maintain system configuration files for the system topology and the user profiles

The TSCV product runs on a CGS workstation, and can execute in two modes: either as a batch program which executes its task solely on basis of data provided as parameters on the command line, or an interactive program.

8.2.2.1 Selected CCU Version

The access ways to data in the configuration database is like a directed graph (with nodes of varying types). The database concept is such that all configuration data prepared for a test must be collected under one and the same access node, which shall be of type <u>CCU</u>. TSCV lets the user select such a node, and uses this selected CCU as a 'root' address in the access tree.

The notion of <u>version</u> is also used in the configuration database, i.e. any node in the access graph may appear in several versions. The user therefore not only must select a CCU node; a specific version of that node must also be selected.

8.2.2.2 Test Configuration

A Test Configuration is a data entity in the configuration database (i.e. an end–item). It is a prescription of how to set–up the CGS for a specific test. Each CCU node may 'own' several such entities. TSCV lets the user select such an entity from the selected CCU. After loading, it resides within TSCV, where it will be used as a template when the user afterwards chooses to execute the Set–up function.

TSCV has space for up to five simultaneously loaded test configurations (which will be displayed in a list and from where the user can select to 'focus' on one of them). This corresponds to concept that the nodes in CGS network can cooperate in groups. Each node on the CGS network may or may not be in one group of nodes which participates to execute a certain test activity. At any time, the CGS network may be divided into 5 cooperating groups (plus the group which consists of the nodes which are not part of any group). The term Test Configuration is used also for these groups. The loaded test configurations can perform testing/test execution concurrently.

The test configurations managed by TSCV are normally selected and loaded from MDB, and are used to set–up the test system to execute testing and record the data acquired during the test into the test result database.

TSCV can also set–up the test system to replay a recorded test session, i.e. data stored in the test result database during a former test session. The test configuration used for replay set–up may be the same as applied during the recording. However, it is also possible to associate an existing session with a test configuration already loaded from the mission database. The replayed data sets are then mapped to replaying test nodes based on the MDB pathname of the recorder and the replayer. A special case is when none of the data sets may be mapped in this way, then a mapping based on table position is applied.

8.2.2.3 Setup

It may or may not be a direct correspondence between a specific group on the CGS network and a Test Configuration that TSCV manages. If a Test Configuration do correspond to a group, it is said to be 'executing'. To bring a loaded Test Configuration into use (i.e. to the state active), the user must command TSCV to execute set–up.

However, before executing Set–up, the user has the option to 'edit' a loaded Test Configuration. It is possible to specify that even though a node appears in the Test Configuration template, it shall not participate in the corresponding group. Further, there are some attributes and parameters associated with the nodes that TSCV uses during the Set–up function that are set when loading a Test Configuration, but which the user can change.

After editing the Test Configuration, the user can give a command to 'Execute Setup'. This implies, among other things, to send a command to the testnodes requesting initialization (the selected test execution mode is sent with this command). It also implies to control initialization of TSS.

Figure 8–1 shows an example of an EGSE set–up with three *Active* Test Configurations (groups).

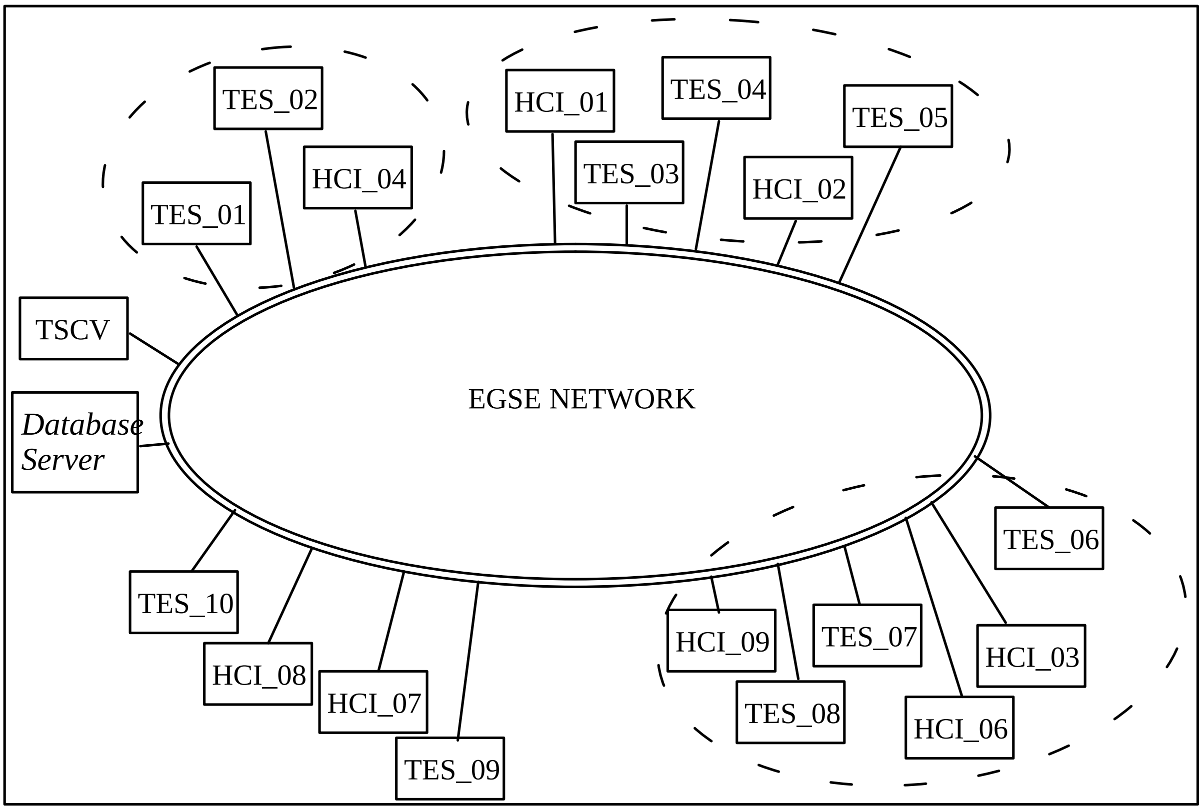


Figure 8–1 : *Example of a set–up with three Active Test Configurations*

8.2.2.4 Testnode Commanding

The user may use TSCV to send commands to the testnodes to start, suspend, resume, and to stop testing. These functions will act upon one testnode or a set of testnodes that the user has selected from one of the Test Configurations.

8.2.2.5 Test Session Create/Open and Close

A Test Execution Session may be regarded as an envelope into which all test data generated during a test are inserted. Before starting testing in one of the groups within the CGS network, TSCV cooperates with the user to create/open a uniquely named test session. Now comes a period in time when the session is open, and test data inserted into this session envelope. Then the user can command TSCV to close the test session. From then on, no data may be inserted into it.

There will be one such session envelope for each Test Configuration which is executing, i.e. for each group on the network. Hence, at any time there may be up to five simultaneously open test sessions in the test result database.

The Test Result Database manager will accept test–data also from a group for which there is no open test session. These data goes into the single Default Session.

8.2.2.5.1 Deleting Test Sessions

The user can select a test session from the Test Result Database and to command the Test Result Database manager to delete that session. The envelope as well as its content will then be deleted. The envelope is represented by the key record created at session opening.

8.2.2.6 System Table Maintenance

TSCV also provides access to a few essential system tables. Functionality is provided to maintain the two tables SYSTEM_TOPOLOGY_TABLE and USER_PROFILES, while a view is provided of the VERSION_ID_TABLE (installed software versions).

8.2.2.7 System Services Management

The test conductor can control a few essential system services with TSCV. It will be ensured that the central DBS applications are up and running at TSCV startup, during setup of a test configuration and when checking status for a test configuration.
Further, the Time System Services which provide the Simulated Mission Time will be started and set–up as necessary.
Also the CGS basic services running on the different CGS computers may be started or shutdown with TSCV.

8.2.2.8 Operations Environment

TSCV's Configuration File

An example of the TSCV configuration file is found in appendix C.

The file is accessed during program initialization when values of various constants are read in. The file is in ASCII format, hence it is possible to read it and to modify it in an ordinary text editor. It contains the identification of a set of parameters and their values, interspersed with comments. If any of the parameters are removed (or name is changed), or if the format of the text which gives the value to the parameters is not as presumed by TSCV, then the file is to be considered as destroyed.

When the configuration file is read, all lines starting with an exclamation mark are skipped; these lines are considered as comment lines.

The parameter definitions consist of three parts, each separated by one or more blanks:
- the parameter name
- a parameter type indicator which may have the value 1, 2 or 3
- the parameter value

Example:
```
TSCV_ICON_FILE_NAME 3 ”tscv_icon.dat”    !File containing a bit map representing the
!                    !icon associated with the TSCV application.
!                    !(To be concatenated with the value of the
!                    !environment variable CGS_TSCV_SAV_DIR)
```

Parameter type 1 is used for integer type parameters. Parameter type 2 is for parameters of type floating point – they must be written in decimal notation and must contain a decimal point. Parameter type 3 is for string type parameters. String parameter values must be enclosed by double quotes, they must have a length of at least one character (e.g. a space for an empty string).

TSCV's Use of Debug Environment Variable

TSCV is 'instrumented' to optionally output debug information. Each procedure within TSCV (except ADT operations) calls a 'debug trace' package when it is called with a string to be output, making it possible to trace the processing. The debug trace package will either output the string or not, depending on the value of an environment variable named DEBUG_TRACING_IS_ON. Only if it is defined and has the value

TRUE will there be produced debug output. If the debug is switched on, there is a set of environment variables which can be used to control the produced output:

- SIZE_OF_DEBUG_BUFFER
  Integer. The trace messages are buffered. Buffer overflow can occur. Default value of this parameter is set in the TSCV configuration file.
- DEBUG_TRACE_ONLY_TO_FILE
  Will have effect if its value is TRUE (i.e. tracing to a separate window is suppressed).
- DEBUG_TRACE_FILE_NAME
  Default value of this parameter is set in the TSCV configuration file.
- DEBUG_EXCLUDE_n (where n is an integer in the range 1 to 5)
  Will suppress printing of strings which commences with the given substring.
- DEBUG_INCLUDE_n (where n is an integer in the range 1 to 5)
  Will override the EXCLUDE setting.

The debug tracing mode will be determined during start–up.

8.2.2.8.1 TSCV Operations Constraints

The TSCV configuration file is stored in *tscv/config*. Data files created/used by TSCV are stored in *tscv/data*. In particular, this directory will contain data which reflect the system state as it was when TSCV was last terminated.

The user who shall run TSCV must have a 'user profile' entry in the user profile database.

The user's profile must give the user one of the following privileges:
– TEST_CONDUCTOR
– TEST_OPERATOR

TSCV cannot execute unless the OpenWindow is running. This applies for the interactive mode as well as for the batch mode.

8.2.2.9 Operation Basics

TSCV provides a set of functions related to administering Test Configurations and set–up CGS according to configuration templates, to monitor system state, to start and stop testing, to open and close test sessions.

TSCV has two command modes:

- The interactive mode where the user commands TSCV interactively by manipulating Graphical User Interface (GUI) items such as buttons, textfields, lists and menus.
- BATCH mode, where TSCV takes as input data provided as command line parameters and executes set–up, starts testing, and terminates when the test is complete.

For reference information on how to use TSCV version 4.1, refer to Chapter 9. Basic interaction techniques are not described, such descriptions may be found in the Open Look documentation.

In the following, an overview presentation is provided of what can be done with and how to operate TSCV. The presentation follows the structure of the main TSCV window. This is horizontally divided in three sections. The upper part relates to the test system as such, i.e. corresponding to basic CGS services. The middle part is where the user mainly will operate. It consists of a list representing the test configurations managed

with TSCV, together with menu buttons representing functions which my be applied on these. The lower part of the window presents details of that test configuration which at any time is selected in the test configuration list, i.e. details relating to each of the applications associated with the test configuration. The lower part has two menu buttons, providing the possibility to control each node separately.

8.2.2.9.1 Test System Control and Configuration

The upper part of the TSCV main window is for the Test System maintenance tasks.

The test system, i.e. the various processes making up the test system, may be started and stopped, and a menu is provided to activate these functions. When activating the Start function, a script which is part of the CGS will be activated. This script will start–up all the CGS services on all the test system computers. The Shutdown function will accordingly invoke a shutdown–script, but before that script is invoked, any test activity will be stopped and resources allocated to any active test will be released.

Certain properties of the test system may be viewed and edited. This is properties represented by the SYSTEM_TOPOLOGY_TABLE, the USER_PROFILES and the VERSION_ID_TABLE. The latter may only be viewed, while the others may be edited.

The SYSTEM_TOPOLOGY_TABLE is a mapping between physical application instances and hosts. Functionality is provided to add, delete and modify application entries. Before making the user's changes applicable, TSCV will check a range of consistency constraints. Each CGS component reads in this table during start–up, thus normally the test system should be stopped and restarted after changes (otherwise applications started before and after the change will know the system topology differently). TSCV automatically provides the user with the option to shutdown the whole system, including TSCV itself.

8.2.2.9.2 Test Configuration Management

The main task for TSCV is to provide control of test configuration set–up. This concept is described in section 4. Up to five test system set–ups (groups of cooperating test resources) may be managed in parallel. Each such set–up is represented by a 'slot' within TSCV. Each slot may be filled with a test configuration table, which serves as a template for set–up. When test system resources are allocated to a slot, it is said to be **Active**. An **Idle** test configuration slot is a template for a potential set–up.

The test configuration slots are represented by a list in the central part of TSCV's main window, which thus has a fixed set of five entries, mapping slots maintained by TSCV. In the list, some key data is provided for each slot, according to the list header: test configuration name, whether the status is **Idle** or **Active**, and the name of a test execution session which may be associated to the slot. In addition, there is space for the name of a session to be replayed – which will be applicable only for replay set–up. When starting TSCV, the same set of test configurations will appear as when last stopped. After installing TSCV, the list will of course be empty. By functions available through the **Edit** menu, the slots may be filled or emptied. Filling a slot implies to load a test configuration (thus, to empty is to **unload**). Normally, load will be from the mission database – corresponding to an option in the **Edit** menu. For replay purpose, it is also possible to fill a slot with a test configuration loaded from the test result database.

All TSCV functions, except for the functions described under sec. 8.2.2.9.1, addresses the test configuration slot which at any time is selected in the list (it is only possible to select one entry at a time).

8.2.2.9.3 Test Configuration Control

Essentially, TSCV's purpose is to set–up test configurations. This means to allocate and prepare resources in the test system to be used for a test, where a test configuration table (together with the SYSTEM_TOPOLOGY_TABLE) is used as a template.

This set–up functionality is available via the menu button **Configuration**. To set–up implies to check that the resources prescribed by the test configuration table are available and not allocated to another test, to allocate the resources and to control the set–up of each individual resource. The following happens when the **Setup** option of the Configuration menu is activated:

- A session is allocated in the test result database. The user may in advance have used the function for creating a session in the slot, in which case TSCV will open that session in the test result database. Otherwise, the database's default session will be allocated. (If already allocated by another test, the set–up will be interrupted). First it is checked whether the central database processes are up and running, and if not, the DBS start–up script will automatically be started.
- The mission database is set–up such that the CCU identifier embedded in the loaded test configuration becomes 'global default' for the test identified by TSCV's slot number.
- The HCIs of the test configuration are allocated, i.e. a message is sent to each application, signalling that it now has become part of an active test configuration. (If already allocated by another test, the set–up will be interrupted).
- The Time Synchronization Services is set–up, i.e. an SMT domain is set–up for the test (the domain will comprise the TSS processes on all hosts affected by the test configuration set–up). First it is checked whether the TSS processes are up and running, and if not the TSS start–up script will automatically be started.
- The TES'es of the test configuration are allocated, i.e. each application instance is commanded to perform initialization. (If already allocated by another test, the set–up will be interrupted). First it is checked whether the TES processes are up and running, and if not, the TES start–up script will automatically be started.
- Optionally, if a replay is prepared, recorded data will be retrieved from the test result database and distributed to the test nodes.

After completion of the set–up, the test configuration and the corresponding TSCV slot is considered **Active**.

To **Start** a test configuration, a corresponding option in the **Configuration** menu should be activated. TSCV will then interact with the allocated test nodes and command them to change from the state **Idle** to **Running**. From that time, they are left to be controlled by HCI. If Start is activated on a test configuration/slot in the state **Idle**, the test configuration set–up will be performed implicitly before sending initializing commands to the test nodes.

To deactivate an active test configuration/slot, the menu option **Stop** should be applied. All test nodes will then be commanded to stop test execution. Furthermore, the test session opened in the test result database will be closed, and the Time Synchronization System will be commanded to clear the allocated SMT domain. All test system resources allocated to the test are released.

A test configuration slot may also be **Shutdown**. By activating this function via the menu, all applications appearing in the slot's test configuration table will be shut–down. If applied on an active test configuration slot, each executing test node will be stopped implicitly before shutting down each participating application. The test session will not be closed, and the test configuration will not changes state. The user will be requested to choose whether or not to include the basic CGS services on the hosts affected by the shutdown (node shutdown).

An active test configuration where one or more of the test nodes are in simulation or replay mode may be commanded to **Suspend** and **Resume** testing. By activating this function with the corresponding menu op-

tions, TSCV will interact with those simulating nodes and command them to change their internal state correspondingly.

TSCV also provides the option to **Check** the status of a test configuration. When the corresponding menu option is activated, TSCV will interrogate the active resources used by the test configuration. If the test configuration state is **Idle**, DBS the TES instances are addressed. Otherwise, also each TSS process allocated to the test configuration will be interrogated. If DBS or TSS does not respond, TSCV will offer the user to start the processes. For the TES instances, only the displayed status will be updated, and should a TES instance be unavailable, the user may launch that individual application and set it up. This is provided via a menu in the lower part of the TSCV main window. By the way, TSCV will periodically and automatically (with a configurable frequency) check the status of the test nodes used by an active test configuration, provided it is in focus.

8.2.2.9.4 Test Configuration Edit

It is the content of the test configuration slots which may be edited – provided that no test system resources have been allocated to the selected slot (i.e. that the slot is **Idle**). There is a separate menu for this.

Normally, a test configuration to be used for set–up will be loaded into the slot from the mission database. It is also possible to set–up such that a formerly recorded test may be replayed. In that case the slot may be loaded with the test configuration entity embedded in the session – which is a copy of the one used during the recording session. There are two corresponding menu options provided. Loading can of course not be done if the selected slot is **Active**. Possible existing data will be overwritten when loading.

Before loading, a test configuration has to be selected – either from the mission database or from the test result database. Thus, the menu options for loading will open separate windows.

The **Load for online test** option is associated with a window labeled **Load Test Configuration for Online Test**, which will list test configuration entities available in the database – for selection and loading. The list contains the entities belonging to one CCU version. There is always a default CCU applicable when the window is opened (normally the same as when the window last was closed). If another shall apply, it must first be selected in another window – opened from the load window. The CCU selected for loading will also apply for the set–up (the test system resources will be informed about the applicable CCU during the set–up process).

The **Load for replay** option is associated with a window labelled **Load Replay Session to be Replayed**. As the window title suggests, loading from the test result database and selection of a session to be replayed is one operation. Thus, in the main window's slot list, the session to be replayed will be identified – and this is the immediate visual feedback that a slot is for replay (regardless the Active/Idle state). TSCV will ensure that the loaded test configuration entity still is existing in the database (otherwise it will not be possible to replay the session). Note, however, that it is not a requirement to replay a test that a frozen version of the test configuration was applied.

Provided the selected slot is not empty, its content – i.e. the test configuration table details – may be viewed. It is also possible to empty a slot – **Unload** a test configuration – provided it is Idle .

8.2.2.9.5 Test Session Control

During a test, all resources allocated to a slot will enter data into the test result database, to be collected into an entity called test session. DBS always provides a default test session, into which the data will go if not a separate session has been allocated (created) for the test.

The user may create and allocate a test execution session with the selected test configuration slot. A menu option which opens a window labelled **Create Session** is provided for this. When creating a session in the

test result database, TSCV will provide DBS with information about which applications are participating in that test, such that it may direct received data to the right session. Accordingly, the **Close** option will deallocate the session from the slot (and DBS will prevent that any more data is entered into it).

TSCV imposes the constraint that the default test session at any time may be allocated to maximum one active test configuration slot. This constraint is checked before setting up a test configuration slot, and before closing a session. There is also a constraint that only active test configurations may allocate a session in the test result database. Thus, the **Create** and **Close** options have a different effect if applied on an Idle slot; they provide information to TSCV which will be used when the slot subsequently is set–up (to actually create the session in the database and allocate it to the slot).

The **Maintain** option provides access to the **Maintain Test Session** window, in which the user can apply various selection criteria to select subsets of sessions from the test result database. The selected sessions are provided in a list, and individual list entries may be deleted. A selected session may also be closed if it has the status open and is not displayed in the main window of TSCV.

8.2.2.9.6 Replay Session Control

Assign/Load Session to a slot

As described in sec. 8.2.2.9.4, a recorded session is implicitly associated with the slot as a replay session, when loading a test configuration from the test result database. It is, however, also possible to make such an association to a slot containing a test configuration loaded from the mission database. A menu option **Assign** is provided for this, and will open a window labelled **Assign Replay Session to Test Configuration** (which is almost identical to the **Load Replay Session to be Replayed** and **Maintain Test Session** windows). The assigned session will appear in the slot list under the column header Replay Session.

Role of Master Test Processor (MTP) during Replay

It is mandatory to have one testnode acting as the MTP during replay mode. The MTP is the only node reading the time control records from archiving files and setting up the SMT as it was during online session.

The MTP is responsible for maintenance of the time base in the CGS system. Therefore it is recommended for a replay session, to have archive data of the MTP. Otherwise it has to be accepted, that no valid SMT replay time is available.

Note: In case no data has been recorded by the MTP during the test, it is possible for the replay session to declare another test node as MTP during the replay session setup.

Select Time Frame

When a replay session is associated to a slot (regardless whether ’loaded’ or ’assigned’), the menu option **Properties** becomes active – in view or edit mode, depending on whether the slot is **Active** or **Idle**. That window will display the time when the recorded session was created and closed. By default, that time frame will be replayed, but in edit mode it is possible to restrict the time interval (by specifying begin and end time). The time base may be switched between Local Time and Simulated Mission Time, and the time frame may be specified by setting the date and time directly. Otherwise, the user events logged during recording is provided in a list, and the begin and end times may be set by selecting entries in that list. The specified time–frame will be sent to the test nodes when the slot is set–up, and TSCV will itself use it when copying data to be replayed from the test result database to the test nodes’ hosts.

The time frame to be replayed can be selected as local time or SMT. The local time selection is to be preferred to the SMT one, as a mapping will be done from the SMT to find the corresponding local time. Especially in the following cases, selection of the time frame in SMT mode is not appropriate :

- The SMT was not running during the test or part of the test.
- The replay start SMT value anticipates any recorded data
- When nothing is to be replayed by one test node (can make sense in a distributed configuration).
- Usually in distributed configuration, as the time is used to synchronise the test nodes between themselves and best synchronisation is achieved with local time.

Speed factor

Another replay property which may be viewed/edited is a speed factor to be applied by the test nodes during replay. It may be set between 0.1 and 100, and will be sent to the test nodes during set–up.

If the speed factor should is set to 2 orhigher, it may happen that CGS is not able to process all data in time. Also the order of processing might not be guaranteed anymore, and synchronisation between test nodes can easily be lost.

8.2.2.9.7 Test Configuration Application Control

The lowest section in the main window of TSCV represents the test system resources associated with the test configuration slot selected in the middle section.

One non–editable textfield displays the pathname of the currently selected test configuration.
Four non–editable text fields display a hierarchical CCU identifier. This is the identifier embedded in the slot's test configuration – which equals the CCU selected by the user in the process of loading from the mission database into the slot.

The scrolling list displays the node entries in the test configuration. When node entries are selected, properties of the first selected one may be viewed in a Properties window. When double clicking on an entry in the list, that entry will be viewed Properties window. The view is depending on whether the test configuration is **Idle** or **Active** it will open in edit or view mode. Actually, if the selected node is a CSS node, the window will always be in view mode. For an HCI, it is possible to specify whether or not the node shall participate, i.e. addressed for set–up. This also applies for a test node, and in addition it is possible to set test execution mode, a forced loading flag, and an MTP flag. When applying changes made for a node, this only affects the test configuration instance loaded into TSCV (which, however, will be saved in the test result database when creating a session), and TSCV will act upon them when setting up the slot. Only one participating test node may have the Master Test Processor (MTP) role. Thus, it will not be possible to set the MTP node non–participating, and the MTP role may only be switched to a participating node.

Please note that TSCV will ensure that it will not be possible to combine test node execution mode Normal and Replay in one slot.

The first column header in the node list is MTP, and one list entry representing a test node will contain a bracket in that column.

If TSCV determines that a node is not set–up as it should, e.g. that the Time Synchronization System is not responding, this will be reflected with an asterisk in the status column.

To find out more about the node status, the function **Check Status** should be invoked, either from the Configuration menu in the upper part of the main window, or from the Node menu above the node list. In the latter case, only those nodes selected in the list will be addressed.

Otherwise, the **Node** menu makes it possible to address individually selected nodes and give commands relating to these. The range of active menu options will depend on the state of the selected nodes, and the kind of node. Most options are only relevant for test nodes: Launch TES, Setup, Start, Suspend, Resume, Stop . The Launch Services option is for starting up all CGS services on the host indirectly designated by the node entry. The Shutdown option is used to shutdown HCI and TES applications, and the CGS services.

8.2.2.10 TSCV Operation Control Procedures And Instructions

8.2.2.10.1 Interactive Mode

When TSCV is started in interactive mode, a graphical user interface is provided.

When TSCV executes in the interactive mode, all error conditions not handled internally will be escalated to the user, and resolved by him. No foreseen error condition shall cause the program to halt. Some errors may however not be resolved by the user of TSCV, but must be resolved by system maintenance, database maintenance etc.

If TSCV for some reason has not been able to perform user authorization (e.g. no user profile could be found for the current UNIX user account) TSCV will terminate (after prompting the user). The user account must be established – with sufficient authority to run TSCV.

The user may optionally specify command line parameters.
Command line parameters that can be used in interactive mode, are described in section 9.4.1.

8.2.2.10.2 Batch Mode

In batch mode, TSCV is started by a single command line. How TSCV shall perform its execution, is specified by the command line options.

When TSCV runs in batch mode, it will communicate a ’result’ as a Unix completion status. It can be read by the program which invoked TSCV to determine the status upon termination. The exit codes are listed in section 9.5.2.

8.2.2.10.3 TSCV Housekeeping

TSCV uses a lock file to prevent that more than one instance is executing at any time. The file is named
tscv.lock
Note that the mere existence of the file does not prevent starting of TSCV. The file also must be regularly updated by a running TSCV.

TSCV uses files to maintain persistent data. Initially, when TSCV is installed, there will be no data files.

For each of the up to five loaded test configurations, TSCV will leave a file containing an image of the test configuration when terminating. The files will have names as
not_inuse_test_configurationx.dat
where x is an integer in the range of 1 to 5. When TSCV has executed a set–up, one of these files will be replaced with a file with a name as
cgs_test_configurationx.dat
where x is an integer in the range of 1 to 5.

When TSCV terminates, it will save information about the current TRDB status on a file named
tscv_saved_test_session_data
This file will be consulted during start–up, and warnings will be issued if it is found that the test session which was open when TSCV last terminated is no longer the one which is open.

When TSCV terminates, it will store information about the state of the testnodes on a file named
tscv_saved_testnode_status.dat
This file will be consulted during start–up, and warnings will be issued if it is found that any of the testnodes reports to be in another state than what was the case when TSCV terminated.

8.2.2.11 TSCV Reference Information

In the following, the windows and user interface functions of TSCV are described and reference information is given for each.

8.2.2.11.1 Help Method

Help is provided for each window component (panel item) of the TSCV dialogues. To get help, the cursor must be positioned at the window component (panel item) for which help is requested. Then the keyboard button labelled **Help** is pushed. The help text will be displayed in a separate window.

To obtain a help summary for the window, the cursor must be positioned inside the window, but not at one of the window components.

If the user want to attach help to another key than the default key for help, the help key may be specified by setting the resource *OpenWindow.KeyboardCommand.Help.*

8.2.2.11.2 Screen Definitions and Operations

This section describes the functionality of the user interface provided when TSCV is invoked in interactive mode. The content of windows, its components and what to be accomplished by the windows' controls, are described.
The section is the basis for the on–line help texts. Therefore, the description has been adapted to such a usage. Figures of TSCV dialogues will not be part of the on–line help output.

8.2.2.11.3 Items and Controls of the TSCV Main Window

8.2.2.11.3.1 General

When TSCV is started in interactive mode, its **'Main Window'** will be opened. The window is divided into three parts (panels), one for the test site, one for test configurations and one to handle the nodes of the currently selected test configuration.

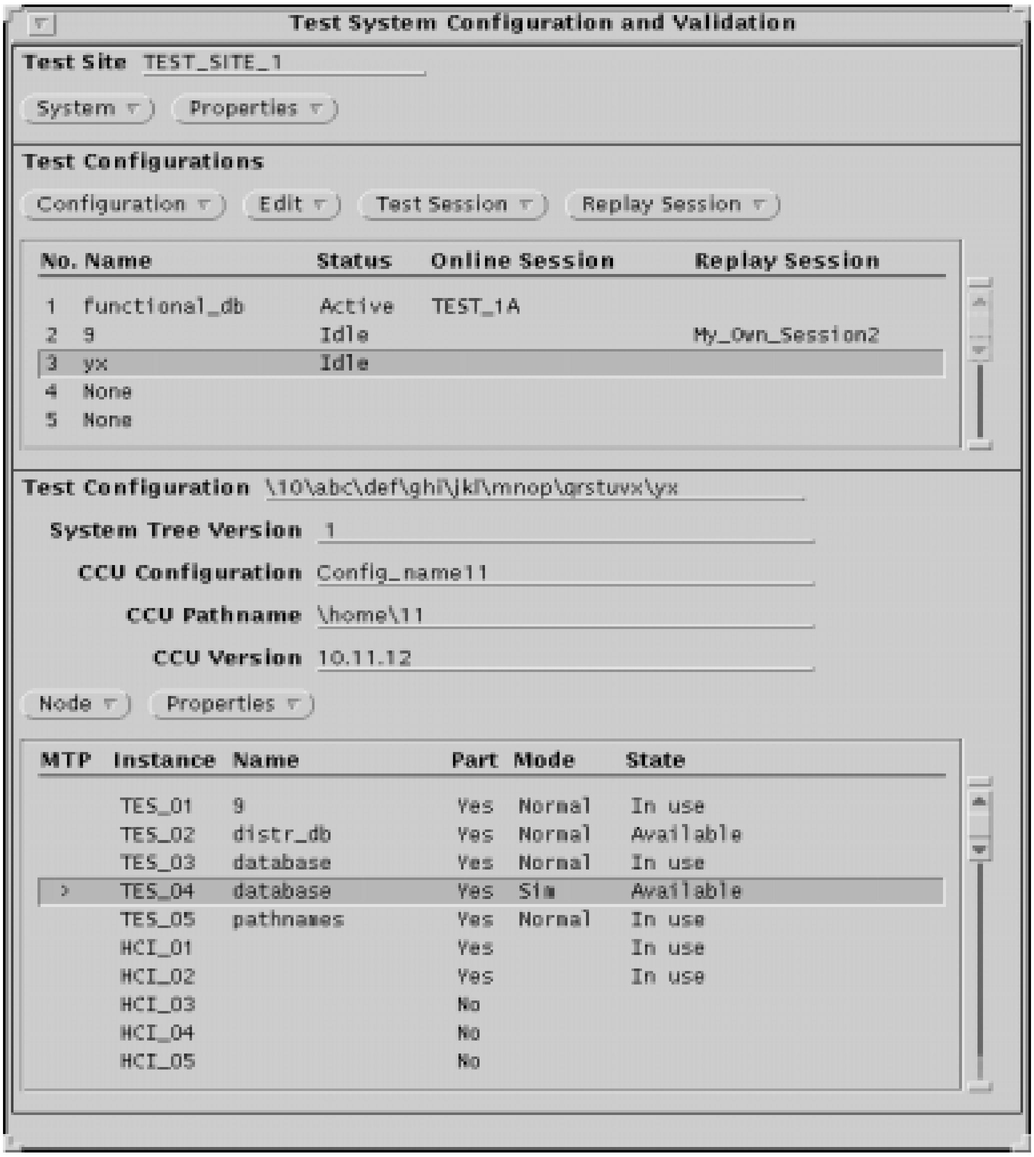


Figure 8–2 : *The TSCV main window*

The **’Main Window’** contains a set of menus used to control the application, and some panel items showing TSCV data and CGS status. Subwindows are opened when the user activates menu items where the label ends with three dots. The subwindows are in general unique for the function associated with the menu item that brings them up. Some exceptions do however exist:

- The **’Select CCU’** window
  The window might be opened in two contexts, one when specifying a CCU as a part of loading a test configuration, and one when specifying the selection criteria filter to list test sessions in the *’Multi Purpose Test Session’* window.
- The **’Multi Purpose Test Session’** window
  To handle selection of test session, one window is used in three different contexts, with a different title an a slightly different content. The three contexts are:
  – Maintain test session, to close sessions, delete sessions and delete from the default session.
  – Load a test configuration from TRDB for replay
  – Assign a test session for replay
- The **’Display Request Window’**
  The window is opened in various contexts when the user requests to view data.

TSCV does not allow parallel execution of operations, i.e. the user can not activate a new operation before the previous has completed. TSCV will use a ”busy pointer” to indicate that an operation is executing.

Many of the operations that the user can activate from the **’Main Window’,** or its subwindows, only apply when TSCV is in a certain state. When the state of TSCV changes, TSCV will set the appropriate panel items active or inactive (dimmed) to give the user a visual feedback of operations which are enabled or disabled. A dimmed panel item means that the functionality connected to the item is not available in the current context.

- The **Test Site** panel
  The panel is related to the test system. It contains the two menus used to control the test system, in addition to a text field displaying the current test site.
  The **’System’** menu, contains the options to launch CGS services and to shutdown the test system.
  The **’Properties’** menu contains the options to configure and view test system data, i.e to open windows in order to maintain the *’System Topology Table’*, to maintain the *’User Profiles’* and to view *’Software Versions’*.

- The **Test Configurations** panel
  The panel is related to loaded test configurations. It displays a list of the loaded test configurations up to the maximum of five, and the menus needed to control the currently selected entry in the list of test configurations.
  The **’Configuration’** menu contains the options to check status, setup, start test, suspend test, resume test, stop test and to shutdown a test configuration.
  The **’Edit’** menu contains the options to load a test configuration from MDB, to load a test configuration from TRDB for replay, to unload a test configuration and to view test configuration data.
  The **’Test Session’** menu contains the options to create, close and delete a test session.
  The **’Replay Session’** menu contains the options to assign for replay a test session to a test configuration loaded from MDB and to open the *’Replay Properties’* window

- The **Selected Test Configuration** panel
  The panel is related to the selected slot in the list of test configurations. For empty slots, nothing is displayed. When a test configuration is selected, its pathname is displayed and its CCU identifier is partly displayed. The nodes associated to the test configuration is displayed in the list of nodes. When doubleclicking one entry in the list, the associated attributes are dis0polayed in the *'Node Property Sheet'* window.
  The menu choices found in the **'Selected Test Configuration'** panel affects only the *selected* nodes in the list of nodes. Two menus are provided:
  The **'Node'** menu contains the options to check status, launch services, launch TES, setup, start test, suspend test, resume test, stop test and to shutdown selected nodes.
  The **'Properties'** menu contain one menu item, **'Node...'** , which displays the attributes of the first selected node in the **'Node Property Sheet'** window.

8.2.2.11.3.2 The 'System' Menu

The menu contains options to control the test system.

8.2.2.11.3.3 The 'System–>Launch Services' Menu Option

The menu option is used to launch the CGS services. It is possible to start the CGS services at any time as long as TSCV is not requested to prevent other user activities. It will not do any harm on applications already running.

8.2.2.11.3.4 The 'System–>Shutdown' Menu Option

The menu option is used to shutdown the test system. It is possible to perform a shutdown of the test system at any time as long as TSCV is not requested to prevent other user activities.

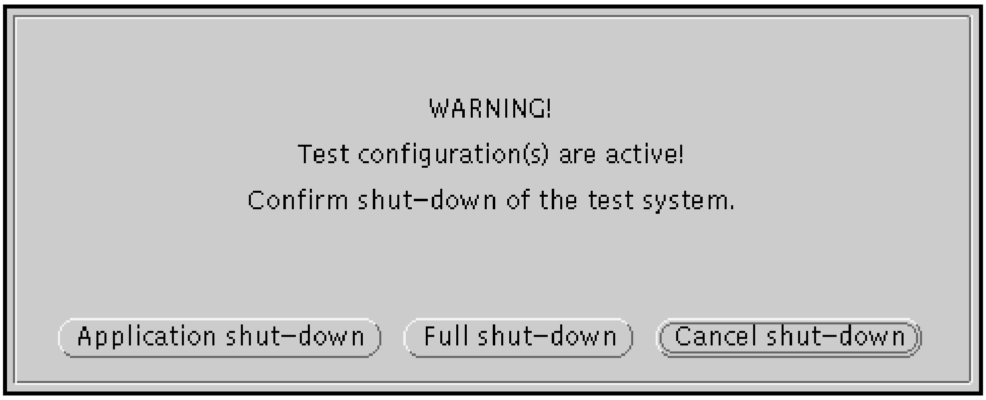


Figure 8–3 : *Confirm shutdown*

Before actually executing the function, the user will be asked to confirm the shutdown request.

If the request is answered by **'Application Shutdown'**, test configurations which have been set–up (is in the state **'Active'**), will be stopped. Stopping a test configuration implies to stop test execution on participating testnodes and close the (possibly) open test execution session.

Then all test and workstation nodes appearing in the *'System Topology Table'* will be commanded to shutdown.

Finally, the *’Central DBS’* will be commanded to shutdown.

If the request is answered by **’Full Shutdown’**, in addition to application shutdown, the remaining CGS applications, e.g. *Network Software* and *TSS*, will be shut down.

If the request is answered by **’Cancel Shutdown’**, shutdown will not be performed.

8.2.2.11.3.5 The ’Properties’ Menu

The menu contains options to maintain the system topology, maintain user profiles and to view software versions.

8.2.2.11.3.6 The ’Properties–>System Topology...’ Menu Option

The menu option is used to open the *’Maintain System Topology’* window, which may be used to view, add, change and remove entries in the *’System Topology Table’*.

8.2.2.11.3.7 The ’Properties–>User Profile...’ Menu Option

The menu option is used to open the *’Maintain User Profile’* window, which may be used to view, add, change and remove user profiles.

8.2.2.11.3.8 The ’Properties–>Software Versions...’ Menu Option

The menu option is used to view the software versions in a display request window.

8.2.2.11.3.9 The ’Configuration’ Menu

The menu contains options to control the test configuration currently selected in the list of test configurations.

8.2.2.11.3.10 The ’Configuration–>Check Status’ Menu Option

The menu option is used to check the status of test nodes and DBS. TSCV prompts the testnodes for their status and renders it in the node list.
When the TSP status inquiry reports an error, the user gets the options to set–up the TSP or to continue. If setup is selected, implicit launch will be performed if the TSP is not responding. Note that TSP status are checked only for active test configurations.

Additionally a window pops up and displays information about the NTP and SMT status of the participating hosts:

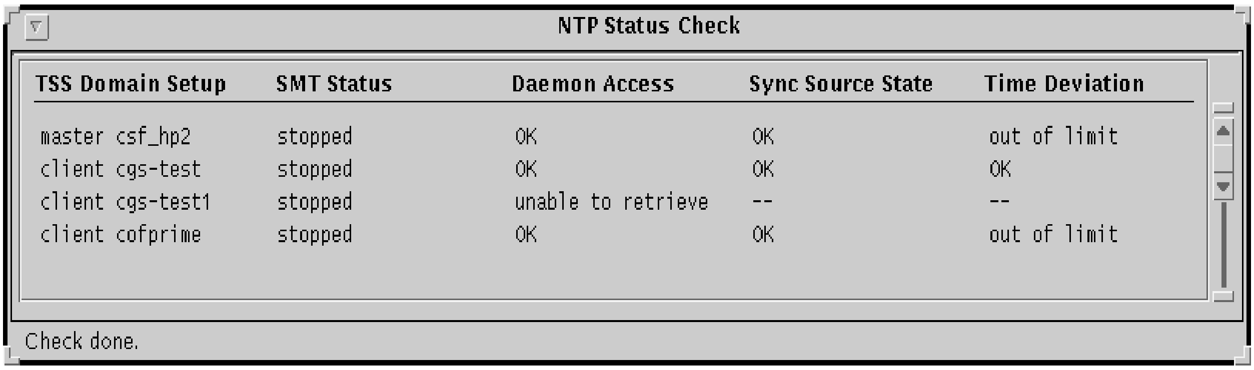


Figure 8–4 : *The NTP Status Check window*

First column ”TSS Domain Setup”:
Shows the participating master and clients of TSS Domain (role + hostname)

Second column ”SMT Status”:
The SMT status of the corresponding host. It can be ’running’ or ’stopped’.

Third column ”Daemon Access”:
It can be ’OK’ or ’unable to retrieve’.
If it is ’unable to retrieve’ verify that the xntp–daemon is running (ps –ef | grep xntpd). If the process is running look for error messages in CGS–Startup outputs.

Fourth column ”Sync Source State”:
It can be ’OK’, ’master unreliable’ or ’—’ (undefined).
If it is ’master unreliable’ the host cannot reach the xntp–daemon on its master host or the master host’s NTP time information are unreliable. If the host is domain master and has no external time source, then there could be problems with the internal hardware clock. If the host ist domain master and has a external time source, then the time source information are unreachable or unreliable.

Fifth column ”Time Deviation”:
It can be ’OK’, ’out of limit’ or ’—’ (undefined).
If it is ’out of limit’ NTP synchronization is running without problems, but the time difference between master and client is greater than allowed.
Reasons for occurence of this status can be a initial installation, configuration changes concerning master–client assignments or a time jump on master side.
In case of initial installation or configuration changes it can take up to 48 hours to sync.

The **’Check Status’** option, also includes check of DBS . If the check concludes that DBS is down, it will be launched.

8.2.2.11.3.11 The ’Configuration–>Setup’ Menu Option

The menu option is used to set–up a test configuration.

TSCV’s actions are determined from the contents of the test configuration currently selected in the list of test configurations.

Initially, TSCV will check that there is at least one participating testnode, i.e. that the ’*participating flag*’ is set for at least one of the testnode entries. Then it will be ensured that the central DBS is up and running (invoking the command to start, if necessary).

At this stage, the status of the test configuration is changed. In the main window, this is recognized by a change in the **’Status’** field in the test configuration list.

Before activating set–up, the user may have associated a test session name with the test configuration (see menu option **’Sessions–>Create’**). If so, TSCV will create a session with that name in the database. If no such session is defined, the default test session will be used. Note that the default test session can only be used by one active test configuration. Set–up will be denied when the default test session already is allocated to an active test configuration.

Next step is to ensure that the TES process on all participating testnodes is up and running (invoking the command to start, if necessary). Then SMT/TSS initialization is performed.

Now the CCU in MDB associated with the test configuration will be set as the global default CCU for the test configuration.

Then the turn has come to set–up the testnodes. This implies to command the TES process on the individual testnodes to execute initiation, i.e. to load configuration data from MDB and enter the execution mode **’Normal’** or **’Simulation’** (as specified in the test configuration).

Finally, the testnodes will be synchronized. If there are more than one participating testnode, TSCV will inform TES on each of the testnodes which other nodes are participating. The TES instances will then exchange configuration information necessary for their cooperation.

The set–up is considered to be executed when TSCV has started setting up the testnodes. If any problems are encountered before that, the function is abandoned and the state of the test configuration remains **’Idle’**. Regardless of any problem occurring during set–up of the testnodes, the result is that the test configuration is in the state **’Active**.

During the setup sequence, progress is reported through a dedicated progress report/abort window. The window contains one button labelled **’Abort’**. Aborting the setup sequence requires a confirmation from the user. When confirmed, the setup sequence will be aborted after end of the operation performing the current setup step.

8.2.2.11.3.12 The ’Configuration–>Start’ Menu Option

The menu option is used to start test execution on a test configuration.

If this option is selected for a test configuration in state **’Idle’**, the set–up function as described for the **’Configuration–>Setup’** menu option will be executed, followed by commanding all participating testnodes to start test execution.

This menu option will also be selectable when the selected test configuration is in the state **’Active’**, and at least one of the testnodes is not in the state **’Executing’**. In this case, each participating and non–executing testnode will be commanded to start test execution.

8.2.2.11.3.13 The ’Configuration–>Suspend’ Menu Option

The menu option is used to suspend testing for test nodes in simulation or replay mode. The menu option will be available for test configurations in state **’Active’** provided that there are executing node(s) of mode **’Sim’** or **’Replay’**. The test execution for all executing test nodes of mode **’Sim’** and **’Replay’**, will be suspended (enters status **’Suspended’**).

8.2.2.11.3.14 The ’Configuration–>Resume’ Menu Option

The menu option is used to resume testing for suspended nodes. The menu option will be available for test configurations in state **’Active’** provided there are node(s) with status **’Suspended’**. All suspended nodes will be resumed, i.e. enter the status **’Executing’**.

8.2.2.11.3.15 The ’Configuration–>Stop’ Menu Option

The menu option is used to *stop* an executing test configuration.

Any testnode which executes testing (status **’Executing’**) will be commanded to stop testing (enter the status **’Available’**).

If there is an open test execution session in the database, this will be closed.

8.2.2.11.3.16 The ’Configuration–>Shutdown’ Menu Option

The menu option is used to shutdown a test configuration, i.e. shutting down all participating nodes of the test configuration. Before actually executing the function, the user will first be asked to confirm the shutdown request.

If the request is answered by **’Application Shutdown’**, and the test configuration is in state **’Idle’**, then all participating test and workstation node are shutdown. If the test configuration is in state **’Active’**, then test execution is stopped on the participating test nodes before the tes tnodes and workstation nodes are shutdown.

If the request is answered by **’Full Shutdown’**, in addition to application shutdown, the CGS services, e.g. Network Software and TSS, will be shut down.

A test configuration shutdown will never change the state of the test configuration.

If the request is answered by **’Cancel Shutdown’**, shutdown will not be performed.

8.2.2.11.3.17 The ’Edit’ Menu

The menu contains options to load, unload and view test configurations.

8.2.2.11.3.18 The ’Edit–>Load for online test...’ Menu Option

When selected, the menu option opens the ’*Load Test Configuration for Online Test’* subwindow, which is used to load a test configuration from MDB into an empty slot or a slot containing an **’Idle’** test configur-ation.

8.2.2.11.3.19 The ’Edit–>Load for Replay...’ Menu Option

When selected, the menu option opens the ’*Load Replay Session’* subwindow, which is used to load a test configuration for replay from TRDB into an empty slot or a slot containing an **’Idle’** test configuration.

8.2.2.11.3.20 The ’Edit–>Unload’ Menu Option

The menu option is used to unload an **’Idle’** test configurations. The test configuration slot will be marked as empty.

If any attributes of the test configuration to be unloaded has changed, a warning is issued and the user has to confirm the unload request.

Figure 8–5 : *Confirm unload*

8.2.2.11.3.21 The ’Edit–>View...’ Menu Option

When the menu option is selected, TSCV will format a textual representation of the attributes associated with the currently selected test configuration in the list of test configurations. The result will be displayed in a text window.
Also when double clicking in the list of test configurations, the selected test configuration will be viewed in a text window.

8.2.2.11.3.22 The ’Test Session’ Menu

The menu contains options to create, close and delete test sessions.

8.2.2.11.3.23 The ’Test Session–>Create’ Menu Option

When the menu option is selected, the session creation window will be opened. The created session will be associated with the test configuration currently selected in the list of test configurations.

The menu option will be dimmed if the database already contains an open test session for the selected test configuration.

8.2.2.11.3.24 The ’Test Session–>Close’ Menu Option

When the menu option is selected, TSCV will close the session associated with the selected test configuration. A window pops up where the status (percent of saved files) of closing is displayed. If the session is closed the window disappears. The user has the option to abort the waiting for closing the test session. Therefore he can press the ’Abort waiting’ button. (Abort waiting do not mean abort closing the session!) If the test configuration does not have an open session, the menu option will be dimmed.

Figure 8–6 : *Close session*

8.2.2.11.3.25 The 'Test Session–>Maintain' Menu Option

When the menu option is selected, the *'Maintain Test Session'* subwindow will be opened.

8.2.2.11.3.26 The 'Replay Session' Menu

The menu contains options to assign test sessions to be replayed and to open a window to view and edit replay properties.

8.2.2.11.3.27 The 'Replay Session–>Assign...' Menu Option

When the menu option is selected, the *'Assign Replay Session to Test Configuration'* window is opened.

8.2.2.11.3.28 The 'Replay Session–>Properties...' Menu Option

When the menu option is selected, the *'Replay Properties '* window is opened. That window is used to view and edit properties of replay test configurations

8.2.2.11.3.29 The Test Configuration List

TSCV can 'manage' up to 5 loaded test configurations. The list shows the test configurations identifier, status and session information. The list entry which is selected represents the test configuration currently 'in focus'.

TSCV maintain a persistent copy of its database; thus, after terminating and restarting TSCV at a later occasion, the test configuration list will again render the same information.

Double clicking in the list will have the same effect as selecting the **'Edit–>View...'** menu choice. TSCV will format a textual representation of the attributes associated with the selected test configuration. The result will be displayed in a text window.

The list of test configurations consist of five columns:

*The **'No.'** column*
This column identifies the test configuration slot number in the range 1 to 5.

*The **'Name'** column*
This column identifies the name of the test configuration, represented by the last part of its path name. When the slot does not contain a test configuration, the **'Name'** column is rendered **'None'.**

The ***'Status'*** *column*
The **'Status'** of the loaded Test Configurations can be **'Idle'** or **'Active'**. Initially, when a test configuration is loaded (read from the mission database), it is **'Idle'**. If the user later commands TSCV to execute set–up, the selected test configuration entry becomes **'Active'**, and will then represent an image of a group of participating, cooperating nodes (workstations and testnodes).

The ***'Online Session'*** *column*
An **'Online Session** 'may be associated with each of the loaded test configurations. While the testnodes within an executing test configuration are processing, testdata are generated. These are stored in the *'Test Result Database'* within an *'envelope'* called a **test session**. The user may associate a session name with a test configuration when it is **'Idle'** as well when it is **'Active'**.

The ***'Replay Session'*** *column*
A **'Replay Session**' may be associated with a test configurations when the recorded data shall be replayed by the loaded test configuration. The test configuration may either have been loaded from MDB or loaded from TRDB (embedded in the test session data).

8.2.2.11.3.30 The 'Test Configuration' Text Field

The text field display the path of the currently selected test configuration.

8.2.2.11.3.31 The 'System Tree Version' Text Field

The textfield displays the **'System Tree Version'** part of the CCU identifier of the selected test configuration. For an empty slot, the text field will be empty.

8.2.2.11.3.32 The 'CCU Configuration' Text Field

The text field displays the **'CCU Configuration'** part of the CCU identifier of the selected test configuration. For an empty slot, the text field will be empty.

8.2.2.11.3.33 The 'CCU Pathname' Text Field

The text field displays the **'CCU Pathname'** part of the CCU identifier of the selected test configuration. For an empty slot, the text field will be empty.

8.2.2.11.3.34 The 'CCU Version' Text Field

The text field displays the **'CCU Version'** part of the CCU identifier of the selected test configuration. For an empty slot, the text field will be empty.

8.2.2.11.3.35 The 'Node' Menu

The menu contains options to control selected nodes.

8.2.2.11.3.36 The 'Node–>Check Status' Menu Option

The menu option is used to update the values of the column **'State'** for test nodes selected in the node list. Hence, the operations initiated will be equal to the operations associated with the **'Configuration–>Check'** menu choice, except that the **'Node–>Check'** menu choice addresses only selected nodes of the test configuration. For initialized nodes, the column **'Mode'** will also be updated according to the acquired status.

The menu option will be dimmed when one or more of the selected nodes are in use by another active test configuration, i.e. the status field is rendered **'In use'**.

This menu option is the only option available if the status of the selected test node is **'Unknown'**.

8.2.2.11.3.37 The 'Node–>Launch Services' Menu Option

By this menu option, the user may initiate the start of the CGS services on the selected nodes.

The menu option will be dimmed if the state of the test node is **'Unknown'**. In this case **'Check Status'** has to be performed before further operations are allowed.

8.2.2.11.3.38 The 'Node–>Launch TES' Menu Option

By this menu option, the user may initiate the launching of the TES process on a node.

The option will be selectable only when the **'State'**, of each entry selected in the node list, are **'No contact'**.

8.2.2.11.3.39 The 'Node–>Setup' Menu Option

The menu option is used to execute set–up on individual test nodes, i.e. to command selected testnodes to load configuration data from the mission database. If the test configuration is in state **'Idle'**, the state will be changed to **'Active'**.

A participating test node will change state from **'Available'** to **'Idle'** when the test node has been commanded to initialize.

This function would be used for instance if one of the testnodes failed to complete its set–up successfully while TSCV executed the set–up function on test configuration level. Another example when this function could be useful was after some kind of crash on one of the testnodes; it would then be possible to launch and subsequentially set–up TES on that testnode only, without performing set–up for the whole test configuration.

This option will be dimmed when one of the selected test nodes have state **'Unknown', 'In use'**, **'Executing'** or **'Error'**.

8.2.2.11.3.40 The 'Node–>Start' Menu Option

This menu option is used to command start testing on individual testnode level. If the test configuration is in state **'Idle'**, setup of the whole test configuration will be performed implicitly.

This option will be dimmed when one of the selected test nodes have state **'Unknown', 'In use'**, **'Executing'** or **'Error'**.

8.2.2.11.3.41 The 'Node–>Suspend' Menu Option

The function associated with this menu option is to command the selected test node(s) to suspend test execution.

This option will only be active when each of the selected test nodes are in the mode **'Simulation'** and the status is **'Executing'**.

8.2.2.11.3.42 The 'Node–>Resume' Menu Option

The menu option is used to command the selected test node(s) to resume test execution.

This option will only be active when each of the selected test nodes are in the mode **’Simulation’** and the status is **’Suspended’**.

8.2.2.11.3.43 The ’Node–>Stop’ Menu Option

The menu option is used to command stop test execution on individual testnode level.

This option will only be active if each of the selected testnodes is in the state **’Executing’**.

8.2.2.11.3.44 The ’Node–>Shutdown’ Menu Option

With this menu option, it is possible to command individually selected test and work–station nodes to shutdown. It is also possible to shutdown the CGS services on the selected nodes.

The menu option will be dimmed if the state of the test node is **’Unknown’**.

8.2.2.11.3.45 The ’Properties’ Menu

The menu option is used to open the *’Node Property Sheet’* window.

8.2.2.11.3.46 The ’Properties–>Node...’ Menu Option

This menu option is used to open the ’Node Property Sheet’ window. When the node list contains multiple selections, then the attributes of the first selected entry will be displayed. A double click at another entry in the list, will display attributes of the double clicked entry.

If the status of the selected test configuration is **’Idle’**, it will be possible to change the value of some of the attributes.

If the status of the selected test configuration is **’Active’**, the window will be ’read only’.

8.2.2.11.3.47 The List of Nodes

The test nodes, work station nodes and the simulation node of the test configuration in focus will be listed here. Please note that when the status of the test configuration is **’Idle’**, then **all** node entries in the test configuration are rendered, while only the **participating** nodes appear when the test configuration is **’Active’**.

As long as the test configuration is **’Idle’**, the **’Node Properties Sheet’** window can be used to edit a subset of attributes for nodes displayed in the node list.

When double clicking in the list, the node property sheet for the selected node is displayed.

The list is formatted in columns with headers **MTP, Instance, Name, Part, Mode and State:**

*The **’MTP’** column*
When more than one testnode is participating, one must have the role **’Master Test Processor’**. Under the column header **’MTP’,** this is indicated by a **’>’** sign in front of the testnode name.

*The **’Instance’** column*
The physical node name is showed in the column **’Instance’**.

*The **’Name’** column*
The logical node name, i.e. the last part of the pathname, is displayed in the column **’Name’.**

The ***'Part'*** *column*
Whether or not the node is/will be participating is indicated by a **'Yes'** or a **'No'** in the column headed by **Part**. When the test configuration is executing, only the participating nodes are rendered, hence only **'Yes'** will be seen.

The ***'Mode'*** *column*
The execution mode associated with the testnode entries in the test configuration is displayed in the column with header **'Mode'**. This mode value will be applied when setting up the test configuration, as well as when setting up individual testnodes. For initiated TES instances (status **Idle**, **Executing** or **Suspended**), the **'Mode'** column renders real mode. Else the **'Mode'** column renders the desired mode as defined/displayed in the node property sheet.

The different mode options possible are as follows:

Normal – This is the normal operational mode where TES is interacting with Special Application Software (SAS).
Sim – In this mode TES does not interact with the Special Application Software. Instead, TES simulates this interaction.
Replay – In this mode TES replays a previous recorded session. Instead of interacting with the Special Application Software, TES "sends" and "receives" the exactly same data as it did when the session was recorded.

The ***'State'*** *column*
The node state as known by TSCV is rendered in the corresponding column. The status is updated as response to user commands; either when issuing a command to set–up, start or stop the test configuration, to set–up, start, suspend, resume or stop individually selected testnodes, or when the **'Check Status'** function is executed.

The different status options possible for test nodes are as follows:

Unknown – The TES instance has never been commanded by TSCV.
No contact – TSCV has no contact with the TES instance.
Error – The TES instance is in an error state.
Available – The TES instance is available and may commanded.
In use – The TES instance is in use by another active test configuration. The instance may not be commanded as part of this test configuration.
Idle – The TES instance has been initialized and is ready to start test execution.
Executing – The TES instance is executing a test.
Suspended – The TES instance has been suspended.

An asterix is appended to the TES statues *'Idle'*, *'Executing'* and *'Suspended'* if synchronization or the Time Service has not been set–up correctly on the node.

For the other instance in the node list, i.e. work–station and simulation nodes, the only possible value shown in the status column is the asterix. This value indicates that the Time Service has not been set–up correctly on the node hosting the instance.

8.2.2.11.4 Items and Controls of the Load Test Configuration Window

8.2.2.11.4.1 General

The **’Load Test Configuration for Online Test’** window will be opened when the main window menu choice **’Edit–>Load for Online Test...’** is activated. The window is used to load a test configuration enditem from the mission database. TSCV has space (’slots’) for up to 5 test configurations, each with a corresponding entry in the test configuration list in the main window. When the **’Load’** button is activated in this sub–window, the selected test configuration will be loaded into the slot currently selected in the main window. Loading a test configuration can also be initiated by double clicking in the list of test configurations.

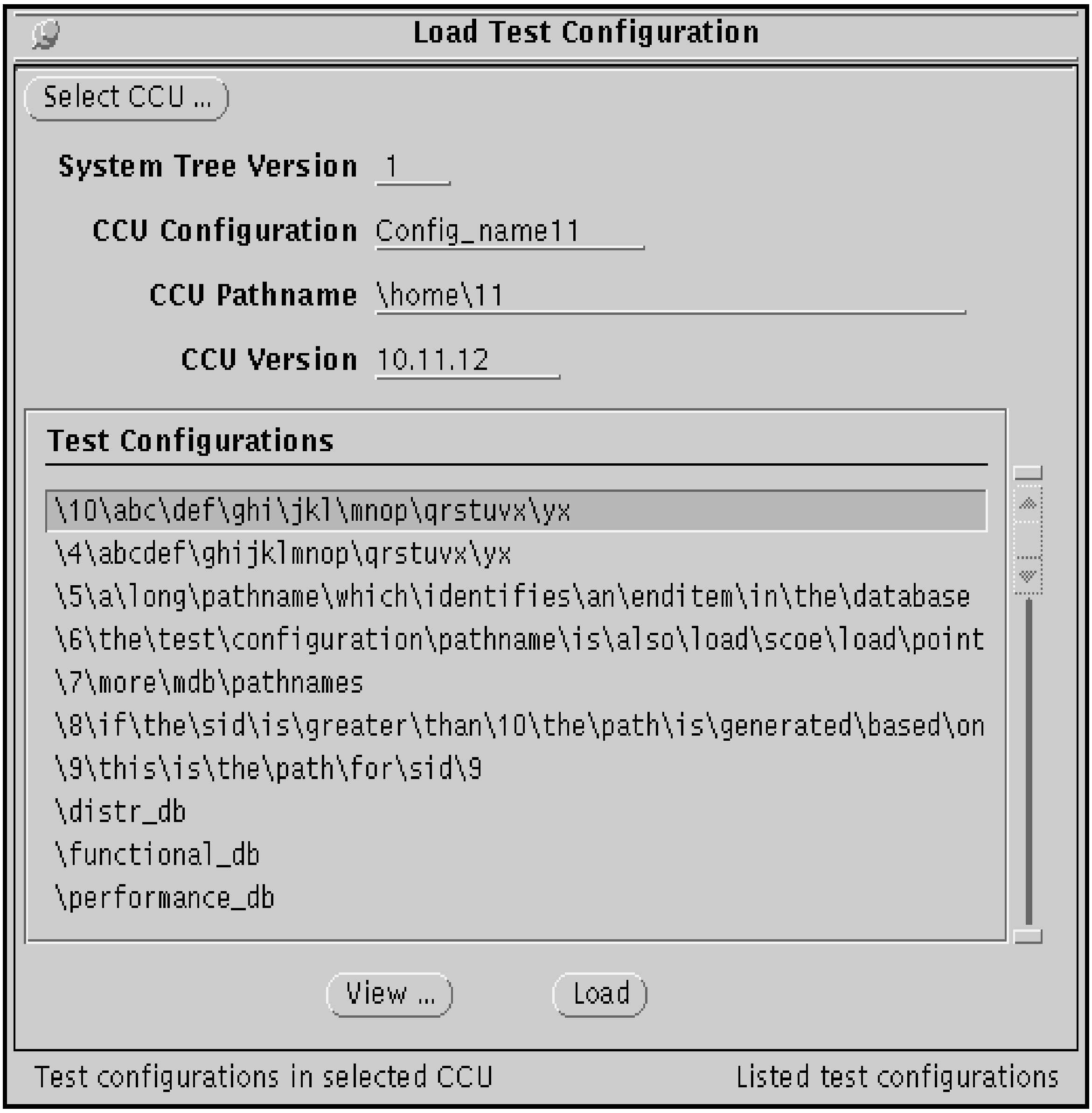


Figure 8–7 : *The Load Test Configuration for Online Test window.*

The window will list all the test configuration enditems in the mission database that ’belongs’ to the CCU viewed in the associated text fields. When the window is opened, the content of the CCU text fields are dependant on the current selection in the list of test configurations of the TSCV main window. If an idle test

configuration is currently selected, the associated CCU will be rendered into the text fields. If the selected slot is empty, the CCU will be the one associated with the last selected test configuration.

If the CCU text fields are empty when the window is opened, it means that TSCV was not able to work out which test configuration that was the last selected one. Note that initially, when TSCV is taken into use in a new environment, all slots will be empty; hence the CCU text fields will also be blank.

The button **'Select CCU ...'** can be used to activate another sub–window for CCU selection. After selection, the list of test configurations will be updated to contain the test configurations that 'belongs' to the selected CCU.

The window lets you view the contents of the test configurations in a text window by selecting the **'View...'** button.

When TSCV has no record of the test configuration last selected, e.g. after the very first invocation of TSCV, then the CCU text fields and the buttons **'View ...'** and **'Load'** will be dimmed. In that case, the user has to activate another sub–window for CCU selection before a test configuration can be selected for loading.

8.2.2.11.4.2 The 'Select CCU...' Button

When the **'Select CCU ...**' button is activated, another window opens, in which a CCU can be selected. The CCU thus selected will be rendered into the associated text fields**.** The list of test configurations will be updated to contain the test configurations that 'belongs' to the selected CCU.

8.2.2.11.4.3 The 'System Tree Version' Text Field

The list of test configurations displays mission database end–items that 'belongs' to a CCU. The text field displays the **'System Tree Version'** part of that CCU identifier.
The text field will be empty and dimmed when a CCU is not identified by TSCV.

8.2.2.11.4.4 The 'CCU Configuration' Text Field

The list of test configurations displays mission database end–items that 'belongs' to a CCU. The text field displays the **'CCU Configuration'** part of that CCU identifier.
The text field will be empty and dimmed when a CCU is not identified by TSCV.

8.2.2.11.4.5 The 'CCU Pathname' Text Field

The list of test configurations displays mission database end–items that 'belongs' to a CCU. The text field displays the **'CCU Pathname'** part of that CCU identifier.
The text field will be empty and dimmed when a CCU is not identified by TSCV.

8.2.2.11.4.6 The 'CCU Version' Text Field

The list of test configurations displays mission database end–items that 'belongs' to a CCU. The text field displays the **'CCU Version'** part of that CCU identifier.
The text field will be empty and dimmed when a CCU is not identified by TSCV.

8.2.2.11.4.7 The 'Test Configurations' List

The list contains the pathname of all test configurations associated with the selected CCU (displayed in the CCU text fields). Whenever a CCU is rendered into the CCU text fields – either when the window is opened

or when a CCU has been selected in the CCU subwindow – TSCV will retrieve all the associated test configurations from the mission database and display their pathname identifier in this list.

When double clicking in the list of test configurations, loading of the test configuration will be initiated.

8.2.2.11.4.8 The 'Load' Button

When the **'Load**' button is activated, TSCV will load the database end–item identified by the currently selected test configuration in the **'Test Configurations'** list. There is space (slots) for up to 5 loaded test configurations, and the retrieved enditem will go into the slot currently selected in the main window. When the loading is completed, this window will be closed, and the main window will be updated to reflect the new test configuration.

If the '**Load'** button is dimmed, this means that no CCU is identified. A CCU has to be selected by activating the button '**Select CCU ...'**

8.2.2.11.4.9 The 'View...' Button

When the **'View ...**' button is activated, test configuration attributes associated with the current selection in the **'Test Configurations'** list, will be formatted for display purposes. A dedicated text window is used to view the attributes.

If the '**View ...'** button is dimmed, it means that no CCU is identified. A CCU has to be selected by activating the button '**Select CCU ...'**

8.2.2.11.5 Items and Controls of the Select CCU Window

8.2.2.11.5.1 General

The **'Select CCU'** window can be used in two contexts:

- **Loading Test Configuration** window:
  The *'Select CCU'* window is opened by pushing the **'Select CCU ...'** button in the *'Load Test Configuration***'** window. It is opened to specify a CCU identifier in the mission database. The selected CCU will be used to retrieve associated test configurations.
  If the *'Load Test Configuration'* window already has a current CCU selection, it will be used as the default selection when the *'Select CCU'* window is opened.
  When a CCU is identified, pushing the **'Apply'** button transfers the CCU into the CCU text field in the *'Load Test Configuration'* window as the 'current CCU'.
  Note that if the **'Select ...'** button in the *'Multi Purpose Test Sessions'* window is pushed, the context of the *'Select CCU'* window will change accordingly
- **Multi Purpose Test Sessions** window:
  The *'Select CCU'* window is opened by pushing the **'Select CCU ...'** button in the *'Multi Purpose Test Sessions'* window. It is opened to select/specify a CCU identifier in the mission database. The selected CCU will be used as filter when a list of test sessions is generated (only associated test sessions will be listed).
  If the *'Multi Purpose Test Session'* window already has a current CCU selection, it will be used as the default selection when the *'Select CCU'* window is opened.
  When a CCU is identified, pushing the **'Apply'** button transfers the CCU into the CCU text fields in the *'Maintain Test Sessions'* window as the 'current CCU'.

Note that if the **’Select CCU...’** button in the *’Load Test Configuration for Online Test’* window is pushed, the context of the *’Select CCU’* window will change accordingly

The **’Select CCU’** window consists of two main parts:

Selection of components in the CCU hierarchy
The selection of a CCU is based on a hierarchical approach. List buttons are used to display a list of components of the corresponding level in the hierarchy. The component collection listed, is based on the selections made for the higher levels in the hierarchy.
The selection of the CCU component at one level in the hierarchy, is done by selecting an item displayed in the list. The list button for the next level in the hierarchy, is made available. By pushing that button, the CCU components based on the current selection, are displayed in the list.
Double clicking in the list, has the same effect as first selecting an item, and then pushing the button to display the next level in the hierarchy.

Current Selection
For each of the CCU components, there is one corresponding text field, where the current selection is displayed. When an item in the currently displayed list is selected, the corresponding text field is updated. Naturally, if one goes back up in hierarchy, the lower level fields will be blanked and dimmed.
The text fields labelled **’CCU Configuration Name’** and **’CCU Pathname’** do not correspond directly to any of the **List** buttons; they are associated with the button **’CCU’**.

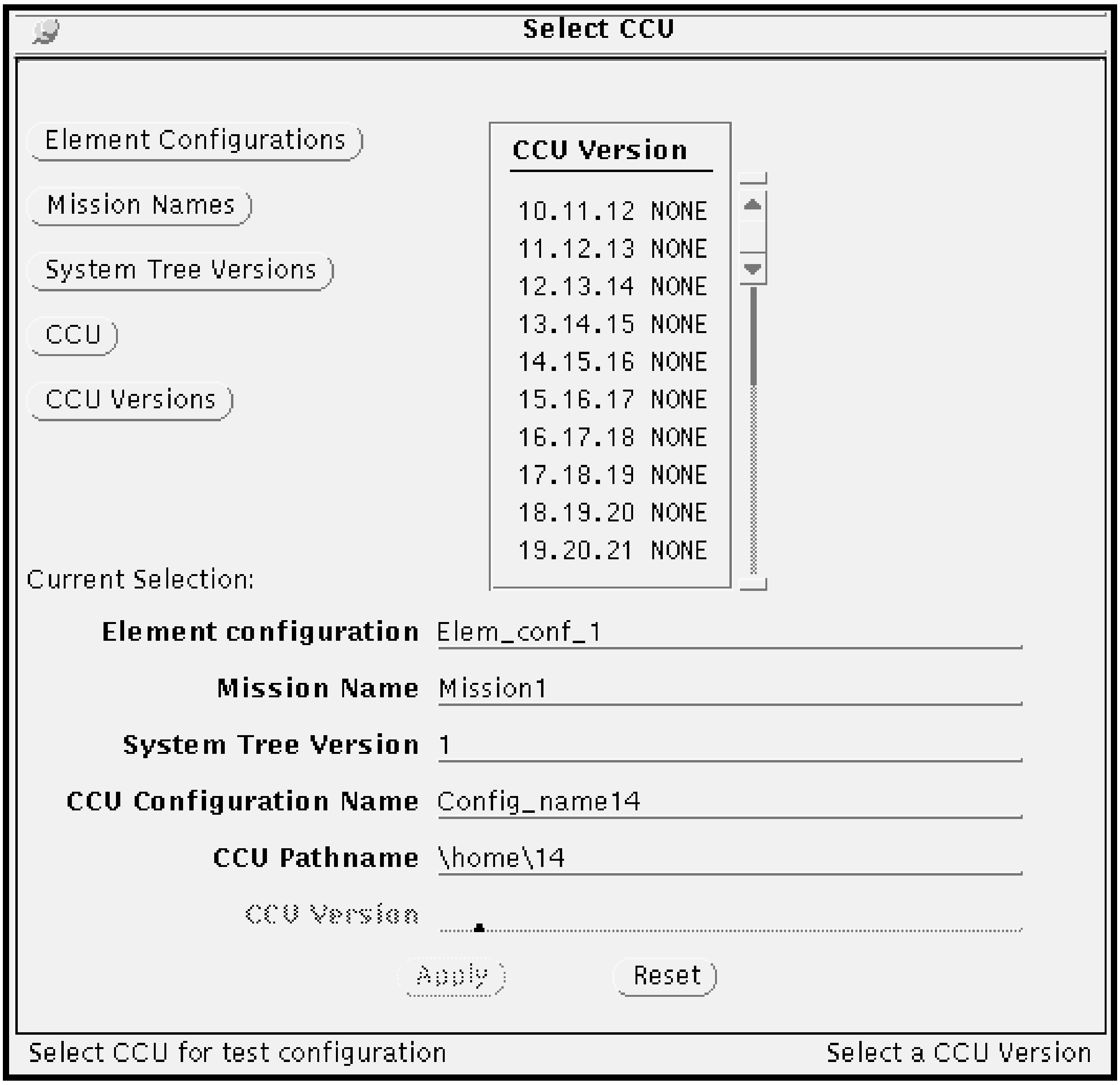


Figure 8–8 : *The Select CCU window*

8.2.2.11.5.2 The ’Element Configurations’ Button

When this button is pushed, the list will display available element configurations. All other buttons, except the **’Reset’** button, will become inactive (dimmed). When selecting one of the items in the list, the **’Mission Names’** button will become active and the selected list item is displayed in the **’Element configuration’** text item for current selections. Double clicking in the list of element configurations will set the double clicked item as selected, list the associated mission names and enable the **’Mission Names’** button.
Note that any selections made, will be lost when pushing the ’**Element Configurations’** button.

8.2.2.11.5.3 The ’Mission Names’ Button

The button is available when an element configuration has been selected, i.e. displayed in the **’Element configuration’** text item for current selections. When the button is pushed, the mission names based on the selected element configuration are displayed in the list. By selecting an item in the list, the **’System Tree Versions’** button will become active and the selected list item is displayed in the **’Mission Name’** text item for

current selections. Double clicking in the list of mission names will set the double clicked item as selected, list the associated system tree versions and enable the **’System Tree versions’** button
Note that when pushing the **’Mission Names’** button, all selections other than the element configuration selection, will be lost.

8.2.2.11.5.4 The ’System Tree Versions’ Button

The button is available when a mission name has been selected, i.e. displayed in the **’Mission Name’** text item for current selections. When the button is pushed, the system tree versions based on the selected element configuration and mission name, are displayed in the list. By selecting an item in the list, the **’CCU nodes’** button will become active and the selected list item is displayed in the **’System Tree Version’** text item for current selections. Double clicking in the list of system tree versions will set the double clicked item as selected, list the associated CCU configuration names and enable the **’CCU Nodes’** button.
Note that when pushing the **’System Tree versions’** button, all selections other than the element configuration and mission name selections, will be lost.

8.2.2.11.5.5 The ’CCU’ Button

The button is available when a system tree version has been selected, i.e. displayed in the **’System Tree Version’** text item for current selections.
When the button is pushed, the CCU configuration names based on the selected element configuration, mission name and system tree version, are displayed in the list. Note that the CCU configuration names not necessarily are unique. By selecting an item in the list, the **’CCU Versions’** button will become active and both the text items **’CCU Configuration Name’** and the **’CCU Pathname’** for current selections, will be filled in. Double clicking in the list of CCU nodes will set the double clicked item as selected, list the associated CCU versions and enable the **’CCU Versions’** button. When this button is activated, the scrolling list will display the CCU configuration names. By selecting an entry in the list, both text fields will be filled.
Note that by pushing the **’CCU Nodes’** button, all selections other than the element configuration, mission name and system tree version selections, will be lost.

8.2.2.11.5.6 The ’CCU Versions’ Button

The button is available when a CCU node has been selected, i.e. displayed in the **’CCU configuration name’** and the **’CCU pathname’** text items for current selections. When the button is pushed, the CCU versions based on the selected element configuration, mission name, system tree version and CCU node, are displayed in the list. By selecting an item in the list, it is displayed in the **’CCU Version’** text items for current selections. Double clicking in the list of CCU versions will have the same effect as a single selection.

8.2.2.11.5.7 The Selection List

There is a button for each level of the hierarchical organized CCU identifier. When any one of these is activated, the mission database will be requested to provide the list of identifier components corresponding to the label of the button (naturally using the already selected identifier components as input), and the received list will be displayed in the scrolling list item. Double clicking in the displayed list will be the same as selecting one item and then push the list button.

When making a selection in the component list, the button for the next lower level of the CCU identifier will be activated.

It is possible at any time to go back up in the hierarchy by activating one of the higher level buttons.

8.2.2.11.5.8 The ’Element Configuration’ Text item

The text item contains the current selection of the element configuration. It is dimmed and cleared to indicate that a selection has not been performed.

8.2.2.11.5.9 The ’Mission Name’ Text item

The text item contains the current selection of the mission name. It is dimmed and cleared to indicate that a selection has not been performed.

8.2.2.11.5.10 The ’System Tree Version’ Text item

The text item contains the current selection of the system tree version. It is dimmed and cleared to indicate that a selection has not been performed.

8.2.2.11.5.11 The ’CCU Configuration Name’ Text item

The text item contains the current selection of the CCU configuration name. It is dimmed and cleared to indicate that a selection has not been performed.

8.2.2.11.5.12 The ’CCU Pathname’ Text item

The text item contains the current selection of the CCU pathname. It is dimmed and cleared to indicate that a selection has not been performed.

8.2.2.11.5.13 The ’CCU Version’ Text item

The text item contains the current selection of the CCU version. It is dimmed and cleared to indicate that a selection has not been performed.

8.2.2.11.5.14 The ’Apply’ Button

When the window was invoked by the ’Multi Purpose Test Session’ window .
When a CCU is identified (all the ’Current Selection’ text items have been filled in), the **’Apply’** button will become active. By pushing **’Apply’**, the selected CCU is made available for the **’Multi Purpose Test Session’** window and its identification thereby displayed in the CCU text item of the window.
The **Select CCU** window is closed.

When the window was invoked by the **’Load Test Configuration’ window.**
When a CCU is identified (all the ’Current Selection’ text items have been filled in), the **’Apply’** button will become active. By pushing **’Apply’**, the selected CCU is made available for the **’Load Test Configuration’** window and its identification thereby displayed in the CCU text item of the window.
The **Select CCU** window is closed.

8.2.2.11.5.15 The ’Reset’ Button

The **’Reset’** button is active when changes have been made in any of the selections. By pushing the button, the CCU identifier initially displayed in the window will be displayed again.
The **’Reset’** button will be dimmed when the select CCU window is opened and after having activated reset. This will then indicate that the window content reflects the initially displayed state of the window.

8.2.2.11.6 Items and Controls of the Create Session Window

8.2.2.11.6.1 General

The **Create Session** window will appear when the main window menu option **'Test Sessions–>Create'** is activated.

Figure 8–9 : *The Create Test Session window*

All testdata generated during a test will go into the same 'envelope' represented by a test session created by the user for that particular test. The **'Create Session'** window is used to create sessions.

It is not mandatory to open a specific session, though. The database will accept test data and event log messages from the participating nodes. The data will go into a 'default test execution session' if there is no other session open.

At any time, there can be up to 5 tests going on in parallel, each with its own session open in the database, either the default session or a named session. Note that the default test session can only be used by one of the test executions.

8.2.2.11.6.2 The 'Session Name' Text item

The name of the session is to be entered into this field. The test session name must be unique. An error message will be issued when trying to create a new session with a name of an existing test session. The test session name can consist of **capital letters**, digits and underscores only. Its length is limited to 20 characters. Example: IF_TEST_3.

8.2.2.11.6.3 The 'Purpose' Text item

This text field corresponds to an attribute which the database will associate with the session. It is simply a descriptive text in free format.

8.2.2.11.6.4 The 'Final Archive Medium' Check box

This check box corresponds to a parameter which will be sent to DBS when the session is opened in the test result database.

If not set, the created test data will only be stored on the harddisk of the database server.

When set, DBS will also copy the received test data to a final archive medium (e.g. optical disc). To identify which archiving device to be used, a special program, the FA_SAS, will be invoked for the purpose. Only one instance of that program is permitted to execute. When busy, this is reported by issuing a warning message.

8.2.2.11.6.5 The 'Apply' Button

When activating the **'Apply'** button, a test session is created according to the specified session name and final archiving selection. After creation of the test session, the window is closed.
The button is dimmed when the **'Session Name'** text field is empty.

The actions initiated when pressing the **'Apply'** button depend on whether the associated test configuration is in the state **Idle** or **Active**

When **Idle**, TSCV will interrogate DBS whether there already exists a session with that name. If not, the session name and the attributes will be stored in TSCV's internal database, and the session name will be displayed in the test configuration list in the main window. Otherwise, the user will be requested to change the session name. Later, when the user activates the Set–up function, TSCV will request DBS to create the session in the test result database.

When **Active,** TSCV will immediately request DBS to create the session in the test result database. Provided that this is successful, the session name will also be displayed in the test configuration list in the main window.

Please note that any modifications made in the window will be completely local to the window until the **Apply** button is pressed.

8.2.2.11.6.6 The 'Reset' Button

The button resets all changes entered in text fields and the' final archive medium' check box. The content of the **'Create Session'** window will be as when opened.

8.2.2.11.7 Items and Controls of the Multi Purpose Test Sessions Window

8.2.2.11.7.1 General

The **Multi Purpose Test Sessions** window, may be opened in three different contexts where the window is named:
– The *'Maintain Test Session'* window
– The *'Load Test Session to be Replayed'* window
– The *'Assign Replay Session to be Replayed'* window

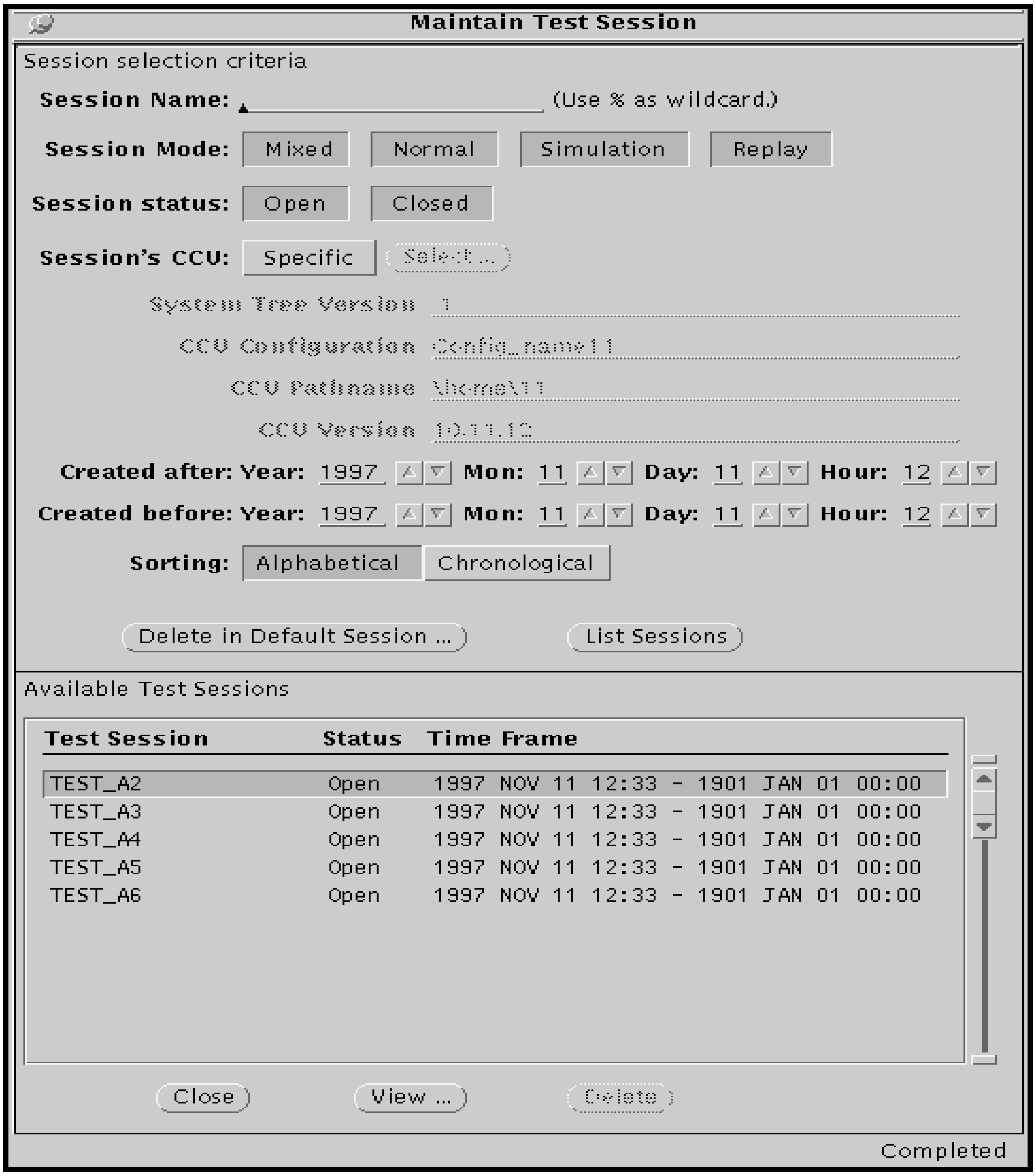


Figure 8–10 : *The Maintain Test Session window*

When the window is opened, the filter criteria will be as when last opened. One exception is the text fields identifying the CCU, which will contain the CCU of the test configuration which is in focus in the main window. The different parts of the window are:

The session list filters
The session list filters are used to restrict which sessions to be retrieved from the test result database. The filter includes specification of the following parameters:
– session pattern
– session mode
– session status
– CCU
– time span (’*created after*’ to ’*created before*’)
– sorting criteria

List sessions
When operating the **'List'** button, the subset of sessions fulfilling the filter parameters, will be displayed in the scrolling list.

Operations on a selected session
Two buttons are available to operate on the session currently selected in the list of sessions. When pushing the **'View...'** button, the key data of the selected session will be viewed in a text window.

The other button, also named the <action> button, will have different labels and and initiate different functions dependent on the context the window is opened in:

The *'Maintain Test Session'* window:
When the **'Delete'** button is activated, the currently selected test session is deleted after user confirmation.

The *'Load Replay Session'* window:
When the **'Load'** button is activated, the embedded data of the currently selected test session is loaded into the currently selected slot in the main window preparing for replay of the recorded data of the test session.

The *'Assign Replay Session'* window:
When the **'Assign'** button is activated, the embedded data of the currently selected test session is assigned to the test configuration currently selected slot in the main window preparing for replay of the recorded data of the test session.

A third button will be available in the *'Maintain Test Session'* window. This button is named **'Close'**, and is used to close sessions open in the Test Result Database, but not known to TSCV. The button will be dimmed when the selected session is listed in the test configuration list or when the session state is different from *'Open'*.

The default session
The window provides one operation on the default session: To delete data in the default session with a time stamp older that the one specified by the **'Created before'** text fields.

8.2.2.11.7.2 The 'Session Pattern' Text item

This part of the session list filter is used to restrict the names of the sessions to be retrieved. The text field is used to specify a session name pattern that has to match for the test sessions to be listed. By specifying the wild–card , '%', there will be no restrictions on the session names.

8.2.2.11.7.3 The 'Session mode' Choice

This part of the session filter is used to restrict the mode of the sessions retrieved to be displayed in the **'Selected Sessions'** list. The choice is non–exclusive.

Mixed
Sessions containing nodes with different modes will be displayed.

Normal
Sessions with session mode **'Normal'** and **'Mixed'** will be displayed. The session mode **'Normal'** implies that the mode of all the TES instances, participating in the test execution, is **'Normal'.**

Simulation
Sessions with session mode **’Simulation’** and **’Mixed’** will be displayed. The session mode **’Simulation’** implies that the mode of all the TES instances, participating in the test execution, is **’Simulation’**.

The *’Delete Test Session’* window
Replay
Sessions with session mode **’Replay’** and **’Mixed’** will be displayed. The session mode **’Replay’** implies that the mode of all the TES instances, participating in the test execution, is **’Replay’.**

8.2.2.11.7.4 The ’Session status’ Choice

The *’Delete Test Session’* window
The choice is non–exclusive and determines the status of the test sessions to be listed.

Open
By selecting the **’Open’** choice, the named sessions which are currently open, will be displayed. TSCV is able to manage up to 5 simultaneously open named sessions (in addition, there will be a default session, which is not displayed in the list).

Closed
By selecting the **’Closed’** choice, the sessions which are currently closed, will be displayed.

The *’Load Replay Session’* window and the *’Assign Replay Session’* window
The choice is always dimmed. It is set to **’Closed’** by default, because data can only be replayed for closed test sessions.

8.2.2.11.7.5 The ’Specific’ Choice

When a session is created in the *Test Result DataBase*, it will be associated with a CCU via the test configuration representing the current test system set–up. It is possible to restrict the listed sessions to those created for a specific CCU. When the **’Specific’** choice is activated, the dimming of the belonging text fields and button will cease. The database will then return only the sessions associated with the CCU identified in the text fields.

8.2.2.11.7.6 The ’Select...’ Button

The **’Select...’** button will become active when the session list filter **’Session’s CCU’** is set to **’Specific’**. By operating the button, the **’Select CCU’** window is opened and that window may be used to select a CCU to be used as a session list filter.

8.2.2.11.7.7 The ’System Tree Version’ Text Field

The text field displays the **’System Tree Version’** part of the currently selected CCU identifier, which may be used to restrict the content of the list. When the text item is dimmed, it indicates that there will be no CCU restrictions as part of the session list filter when operating the **’List’** button.

8.2.2.11.7.8 The ’CCU Configuration’ Text Field

The text field displays the **’CCU Configuration’** part of the currently selected CCU identifier, which may be used to restrict the content of the list. When the text item is dimmed, it indicates that there will be no CCU restrictions as part of the session list filter when operating the **’List’** button.

8.2.2.11.7.9 The ’CCU Pathname’ Text Field

The text field displays the **’CCU Pathname’** part of the currently selected CCU identifier, which may be used to restrict the content of the list. When the text item is dimmed, it indicates that there will be no CCU restrictions as part of the session list filter when operating the **’List’** button.

8.2.2.11.7.10 The ’CCU Version’ Text Field

The text field displays the **’CCU Version’** part of the currently selected CCU identifier, which may be used to restrict the content of the list. When the text item is dimmed, it indicates that there will be no CCU restrictions as part of the session list filter when operating the **’List’** button.

8.2.2.11.7.11 The ’Created after’ Timestamp

When the window is opened, the **’Created after’** timestamp is set to beginning of 1995, i.e. no time restriction. It is possible to restrict the time span with the individual numeric text fields for year, month, day and hour. Only sessions created after the indicated point in time, will be displayed in the list.

8.2.2.11.7.12 The ’Created before’ Timestamp

When the window is opened, the **’Created before’** timestamp is set to the moment when the window was opened i.e. no time restriction. It is possible to restrict the time span with the individual numeric text fields for year, month, day and hour. Only sessions created before the indicated point in time, will be displayed in the list.

Please note that the point in time specified for **’Created before’** will also apply when deleting data from the default test execution session.

8.2.2.11.7.13 The ’Sorting’ Choice

The choice is used to determine sorting of the test sessions to be listed. There are only two options, **’Alphabetical’** or **’Chronological’.**

8.2.2.11.7.14 The ’List’ Button

When the **’List’** button is pressed, the selection filter will be submitted to the database, which will return a list of sessions. The content of the scrolling session list will be replaced by the acquired session list.

8.2.2.11.7.15 The ’Delete in Default Session...’ Button

When the **‘Delete in Default Session...’** button is pressed, TSCV will take the time specified by the **’Created before**’ items, format it and display it in a confirm window. The user will be requested to confirm the wish to delete old data acquired into the default test execution session – data acquired before that point in time. The request will then be submitted to the database for execution.

If one does not use always named sessions, but (as an implicit result of this) the default test session instead, it is possible that the reserved table space for the default test session becomes completely occupied. It is thus necessary to delete the entries in the default test session from time to time. To achieve this, use the **Sessions–>Maintain...–>Delete in Default Session...** *function on a regular basis. Otherwise, DBS will report database (Oracle) errors when the event table becomes full.*

8.2.2.11.7.16 The ‘Test Sessions’ List

The scrolling list in this window will display sessions retrieved from the test result database when the button labelled **’List’** is pressed. Usually, it is required to select a restricted set of sessions.
TSCV will provide a selection filter to the database, and the selection will be restricted by the values specified for the filter items.

8.2.2.11.7.17 The ’View...’ Button

For each test execution session, the database keeps a key record (’session info record’). By pressing the **’View...’** button, the session key data of the session currently selected in the **’Select sessions’** list , will be displayed in a separate text window.
The button will be dimmed when the **’Select sessions’** list is empty.

8.2.2.11.7.18 The <action> Button

The <action> button is labelled dependent on the window context. The associated operations to be performed, will also different.

<u>The *’Delete Test Session’* window, button **’Delete’**</u>

When the **‘Delete’** button is pressed, it is a request to delete the session currently selected in the **’Select Sessions’** list. Before deleting the session, the user is asked to confirm the delete request. When the request is acknowledged, the complete ’session envelope’ (the session key record and all test data acquired into it) identified by the selected session name, is deleted.
The button will be dimmed when the **’Select Sessions’** list is empty.

<u>The *’Load Replay Session’* window, button **’Load’**</u>

When the **‘Load’** button is pressed, the embedded data of the test session currently selected is used to load a test configuration into the currently selected slot in the list of test configurations. It is checked that MDB still contain the test configuration embedded in the test session. The recorded data will be replayed on the same nodes as when the data was recorded. The nodes will be set to participating and mode **’Replay’**. Nodes that could not be mapped, is set to non–participating and mode **’Simulation’**. For such nodes, it will later not be permitted to change mode to **’Replay’**.

If the currently selected slot in the list of test configurations contain an **Active** test configuration , the button will be dimmed. Also, the button will be dimmed when the **’Select Sessions’** list is empty.

<u>The *’Assign Replay Session’* window, button **’Assign’**</u>

When the **’Assign’** button is pressed, the currently selected test session is assigned the **’Idle’** test configuration loaded from MDB, which is the currently selected one in the list of test configurations in the TSCV main window. A mapping is performed, where nodes found in the assigned session for replay is mapped to the same nodes in the test configuration loaded from MDB identified by their pathnames. If no match is found, positional mapping is performed after confirmed by the user. Mapped nodes are set participating and in mode **’Replay’.** For nodes where no replay data could be mapped, the node is set to non–participating and the mode is set to **’Simulation’**. For such nodes, it will later not be permitted to change mode to **’Replay’**.

If the currently selected slot in the list of test configurations do not contain an **Idle** test configuration loaded from MDB, the button will be dimmed. Also, the button will be dimmed when the **’Select sessions’** list is empty.

8.2.2.11.7.19 The ’Close’ Button

This button is used to close sessions open in the Test Result Database, but not known to TSCV. The button will be dimmed when the selected session is listed in the test configuration list or when the session state is different from *’Open’*.

8.2.2.11.8 The Node Property Sheet Window

8.2.2.11.8.1 General

The **’Node Property Sheet’** window will be opened when the the **’Properties–>Node...’** menu choice is activated or when double clicking in the node list in the main window. Note that when more than one node is selected in the list of nodes, properties of the first selected node will be displayed when selecting the **’Properties–>Node...’** menu choice.
The **’Node Property Sheet’** contains entries for a set of attributes related to one single selected node, and which will be acted upon by TSCV when the user commands TSCV to execute the set–up function.

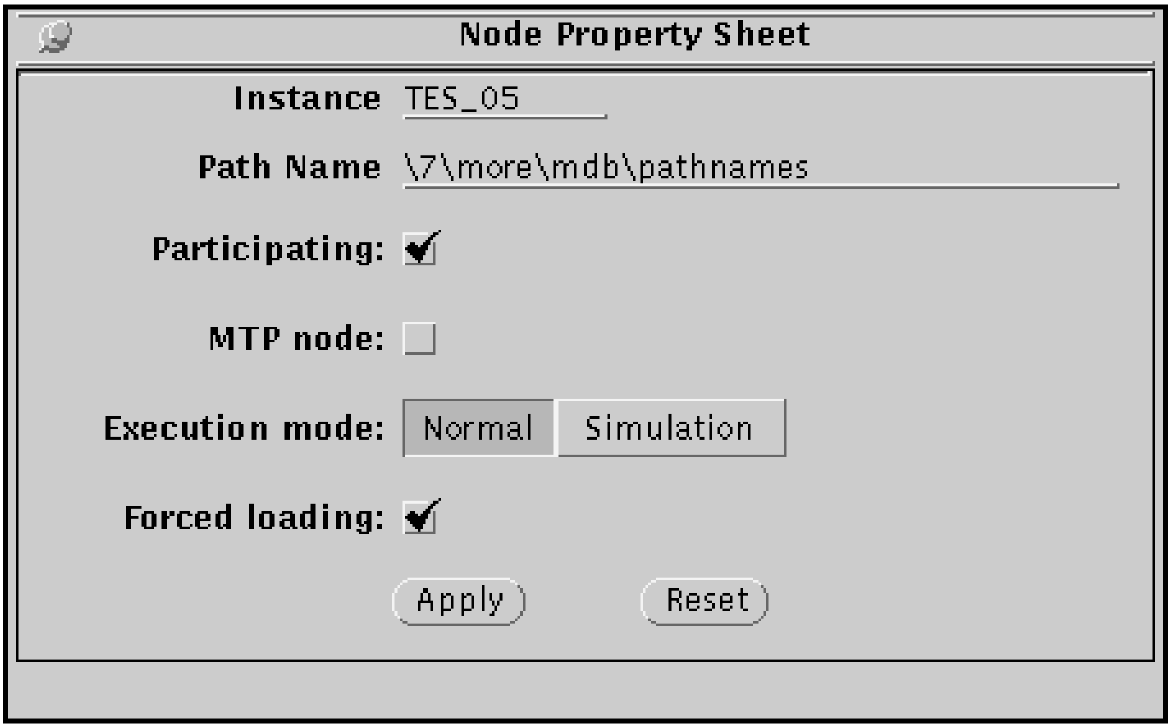


Figure 8–11 : *The Node Property Sheet window*

In the **’Node Property Sheet’** the user can:

- Edit the node attributes for the nodes of an **’Idle’** test configuration (one node entry at a time).
- View the attributes for the participating nodes of an **’Active’** test configuration (one node entry at a time)

Hence, the **'Node Property Sheet'** will open in read only modus (with dimmed window components) when the test configuration in focus is **'Active'**, and in edit modus when the status **'Idle'**.

Both test nodes (TES), workstation nodes (HCI) and simulation nodes (CSS) can be viewed. Only the 'Participating flag' can be edited for HCI nodes, i.e. the rest of the attributes will be dimmed. The CSS node can only be viewed, i.e. no attributes can be changed. For TES nodes, all attributes used in the current CGS version, will be available.

Please note that any modifications made in the window will be completely local to the window until the **Apply** button is pressed.

8.2.2.11.8.2 The 'Instance' Text field

The text field contains the physical name of the node. The attributes of that node is viewed in the **'Node Property Sheet'**. The text field is not editable, i.e. attributes for another node can not be viewed by entering a new name into the text field.

8.2.2.11.8.3 The 'Path Name' Text field

The text field contains the full path name of the node. The text field is not editable.

8.2.2.11.8.4 The 'Participating' Check box

The **'Participating'** checkbox is used to determine whether or not the selected node shall participate in the test when the **'Idle'** test configuration is activated (setup is performed). The **'Participating'** flag can be set for test nodes and workstation nodes. It is dimmed when the node is participating in an **'Active'** test configuration and when the selected node is the MTP node (which can not be made non–participating).

8.2.2.11.8.5 The 'MTP node' Check box

One of the participating nodes of a test configuration must play the role of the Master Test Processor (MTP). Only participating test nodes can be selected as the single MTP and only one of the participating test nodes can be the MTP. When selecting a new node as the MTP node of the test configuration, the previously selected node will automatically be deselected. No actions from the user is required to ensure that only one test node is selected as the MTP node.
When trying to set a non–participating test node as the MTP by pressing the **Apply** button, it will be denied without any notice to the user.

The **MTP** check box is applicable for test nodes only, and will be dimmed for other types of nodes, for the MTP node and when the node is participating in an **Active** test configuration (then the MTP status cannot be changed).

8.2.2.11.8.6 The 'Execution Mode' Check box – Normal or Simulation

The **Execution Mode** check box is used to define the execution mode of the test node. The test node can execute in the modes **Normal** or **Simulation**.

The **'Execution Mode'** check box is applicable for test nodes only, and will be dimmed for other types of nodes and when the test node is participating in an **Active** test configuration (then changing properties is not allowed)

8.2.2.11.8.7 The 'Execution Mode' Check box – Replay TCs

The **'Execution Mode'** check box is used to define the execution mode of the test node. The test node can execute in the modes **Replay** or **Simulation**. If there is not connected any replay data to the node, the only valid mode is **Simulation**. Then the check–box is dimmed.

The **’Execution Mode’** check box is applicable for test nodes only, and will be dimmed for other types of nodes and when the test node is participating in an **Active** test configuration (then changing properties is not allowed)

8.2.2.11.8.8 The ’Forced loading’ Check box

The **Forced loading** check box is used to determine whether or not loading of the database is required when the test node is initiated (set up). This is required whenever the scope of the data to be loaded has changed (CCU or data load point).

The **Forced loading** check box is applicable for test nodes only, and will be dimmed for other types of nodes and when the test node is participating in an **Active** test configuration (then changing the property is not legal)

8.2.2.11.8.9 The ’Apply’ Button

When activating the **’Apply’** button, the content of the **’Node Property Sheet’** is applied as properties for the selected node.

Please note that any modifications made in the window will be completely local to the window until the **Apply** button is pressed.

8.2.2.11.8.10 The ’Reset’ Button

The button resets all changes. The content of the **‘Node Property Sheet‘** window will be as when opened. The viewed attributes will then display the values currently applied on the selected node.

8.2.2.11.9 The Replay Properties Window

The **’Replay Properties’** window will be opened when the menu option **’Replay–>Properties’** is activated. It contains entries for a set of attributes related to a replay set–up, and which will be acted upon by TSCV when you activates the set–up function.

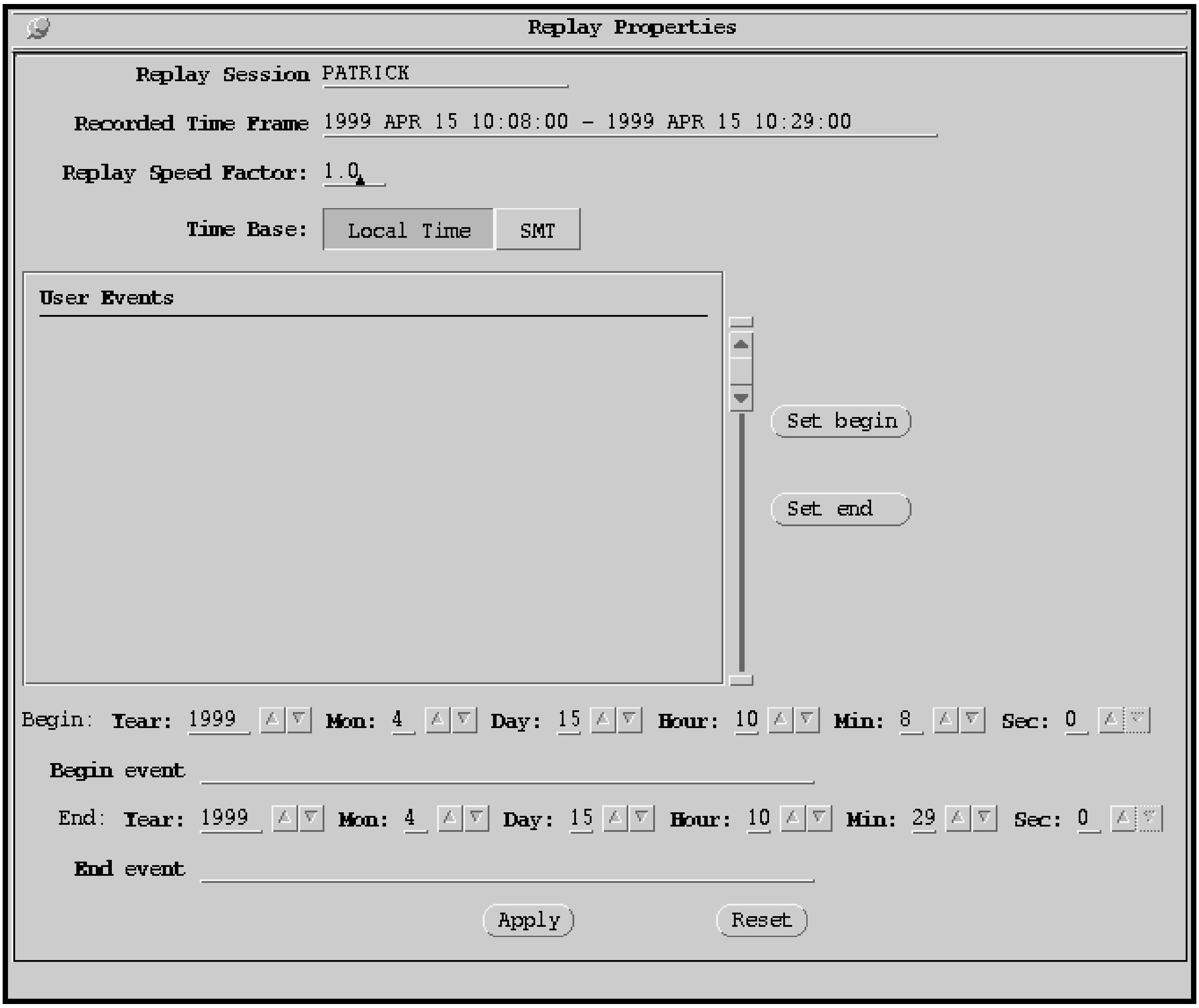


Figure 8–12 : *The Replay Properties window*

In the *’Replay Properties’* window you can:

– See which session has been attached to the test configuration slot for replay purpose.
– See the total available time span (in local time) for the session to be replayed.
– See the user events from the session to be replayed.
– Set replay speed factor.
– Set replay begin and end times by selecting user events.
– Set replay begin and end times in numeric fields representing timestamps.
– Switch between Local Time and Simulated Mission Time representation.

The *'Replay Properties'* window can only be opened when a replay session has been attached to the selected test configuration. It will be opened in read only mode (with dimmed window components) when the test configuration in focus is **'Active'.**

When the *'Replay Properties'* window is open, its contents will be updated when you set another test configuration in focus. Any changes you made, which you have not confirmed by pressing the **'Apply'** button, will then be lost.

8.2.2.11.9.1 The 'Replay Session' Text Field

The textfield contains the name of the recorded session which you have selected and attached to the test configuration in focus as a session to be replayed. The text field is not editable.

8.2.2.11.9.2 The 'Recorded Time Frame' Text Field

The textfield contains the available time frame of the recorded session which you have selected and attached to the test configuration in focus as a session to be replayed, i.e. the session creation time and the session close time. The text field is not editable.

8.2.2.11.9.3 The 'Replay Speed Factor' Text Field

The textfield contains the speed factor which the *Test Execution Software* shall use while replaying data from the session which you have selected and attached to the test configuration in focus as a session to be replayed.

You may specify the value in the range from 0.1 to 100.0. A range and format check will be performed when you hits the **'Enter'** key on your keyboard, and when you activate the **'Apply'** button.

8.2.2.11.9.4 The 'Time Base' Choice

You should use this choice item to switch the time base for the replay time frame.

The specified value (**Local Time** or **Simulated Mission Time**) will be used by TSCV when setting up the test configuration. It will be used as part of a search key when retrieving replay data (archive files) to be provided to the *Test Execution Software*.

The setting will also be reflected in the **'Replay Properties Sheet'** itself: When you select a user event either as a begin time or an end time, the associated time–stamp will be filled in into the corresponding numeric fields. Depending on the current time base setting, either the associated **Local Time** or the **Simulated Mission Time** will be shown.

If the current **Begin Time** or/and **End Time** is/are set via a selected user event (without subsequent editing of the numeric fields), the displayed time–stamp(s) will change and display the event's time in the correct time base.

8.2.2.11.9.5 The 'User Events' List

The list contains the user events belonging to the test session which you have selected and attached to the test configuration in focus as session to be replayed. The events' *'Short Text'* and hour:minute part of the timestamp is viewed in the list (the clock reading be taken from **Local Time** or **Simulated Mission Time**

– according to the current setting).

When you activate either the button **’Set begin’** or **’Set end’**, the point in time when the event was created will be taken as replay begin or end time, correspondingly. The event’s *’Long Text’* will be entered into the corresponding text field, and the event’s timestamp will be entered into the corresponding numeric fields. Depending on the time base setting, the time shown will be either the Local Time or the Simulated Mission Time reading when the event was created.

8.2.2.11.9.6 The ’Set Begin’ Button

You should use this button to specify a user event representing the replay begin time, i.e. the oldest recorded and archived data which will be replayed by the *Test Execution Software*.

When you activate the button, the currently selected event will be taken to represent the begin time. The event’s *’Long Text’* will be entered into the **’Begin event’** text field, and the event’s time–stamp will be entered into the ’Begin time’ numeric fields. Depending on the time base setting, the time shown will be either the **Local Time** or the **Simulated Mission Time** reading when the event was created.

8.2.2.11.9.7 The ’Set End’ Button

You should use this button to specify a user event representing the replay end time, i.e. the latest recorded and archived data which will be replayed by the Test Execution Software.

When you activate the button, the currently selected event will be taken to represent the end time. The event’s **’Long Text’** will be entered into the **’End event’** text field, and the event’s time–stamp will be entered into the **’End time’** numeric fields. Depending on the time base setting, the time shown will be either the **Local Time** or the **Simulated Mission Time** reading when the event was created.

8.2.2.11.9.8 The ’Begin’ Timestamp

The numeric fields contain the current **’Begin Time’**, i.e. the time–stamp from when the *Test Execution Software* will replay the archived data.

When you edit the **’Begin Time’** by modifying one of the numeric fields, the possible association to a selected user event is lost (reflected by the clearing of the long event text).

Each time you change value in one of the **’Begin Time’** numeric fields, a validity check is made. If the time would become greater than the current **’End Time’**, the change will be ignored. If the timebase is **’Local Time’**, and the time would be less than the recorded session’s creation time, the change would be ignored.

8.2.2.11.9.9 The Begin ’Event’ Text Field

This text field will contain the long text of the user event you have selected to reptresent the replay **’Begin Time’**. If you set or modify the **’Begin Time’** by changing value in one of the **’Begin Time’** numeric fields, the association to the selected user event is lost, and consequently the text field is cleared.

8.2.2.11.9.10 The ’End’ Timestamp

The numeric fields contain the current **’End Time’**, i.e. the time–stamp up till when the Test Execution Software will replay the archived data.

When you edit the **’End Time’** by modifying one of the numeric fields, the possible association to a selected user event is lost (reflected by the clearing of the long event text).

Each time you change value in one of the **’End Time’** numeric fields, a validity check is made. If the time would become less than the current **’Begin Time’**, the change will be ignored. If the timebase is **’Local Time’**, and the time would be greater than the recorded session’s close time, the change will be ignored.

8.2.2.11.9.11 The End ’Event’ Text Field

This text field will contain the long text of the user event currently selected to reptresent the replay **’End Time’**. If you set or modify the **’End Time’** by changing value in one of the **’End Time’** numeric fields, the association to the selected user event is lost, and consequently the text field is cleared.

8.2.2.11.9.12 The ’Apply’ Button

When you activate this button, settings you have made in the window will be saved.

8.2.2.11.9.13 The ’Reset’ Button

You may abandon changes you have made in the window by activating this button. The values will be reset to the settings valid last time you activated the **’Apply’** button (or when the recorded session was attached to the test configuration slot).

8.2.2.11.10 The Maintain System Topology Window

8.2.2.11.10.1 General

When the *’Maintain System Topology’* window is opened and after **‘Reset‘,** the *’System Topology Table’* residing at *’$CGS_HOME/config’* will be loaded and all entries put into the list of *’System Topology Table’* entries.

The user may then update the content of the list by first entering values into the text fields **’Host’**, **’Instance’** and **’Port Number’** (only for SAS) and then request to add an entry or change the currently selected entry. The currently selected entry can also be deleted.
Please note that any modifications made in the window will be completely local to the window until the **’Apply’** button is pressed. Also note, that when selecting an entry in the list, the content of the newly selected item will be displayed in the text fields **’Host’**, **’Instance’** and **’Port Number’,** and any changes made in the text items will be overwritten and lost.

When the user wants to update the *’System Topology Table’* of the test system, the **’Apply’** button must be pushed. The consistency of the content of the list of *’System Topology Table’* entries, is checked. When errors are found, this is reported by error messages and applying changes will be denied. When the consistency check is successful, the user can select either to **’Store and shutdown’**, **’Store’** or **’Cancel’**.

The applications, including TSCV, loads the *System Topology Table* when invoked. Therefore, a test system shutdown has to be performed to ensure that changes will be read by the applications throughout the test system.

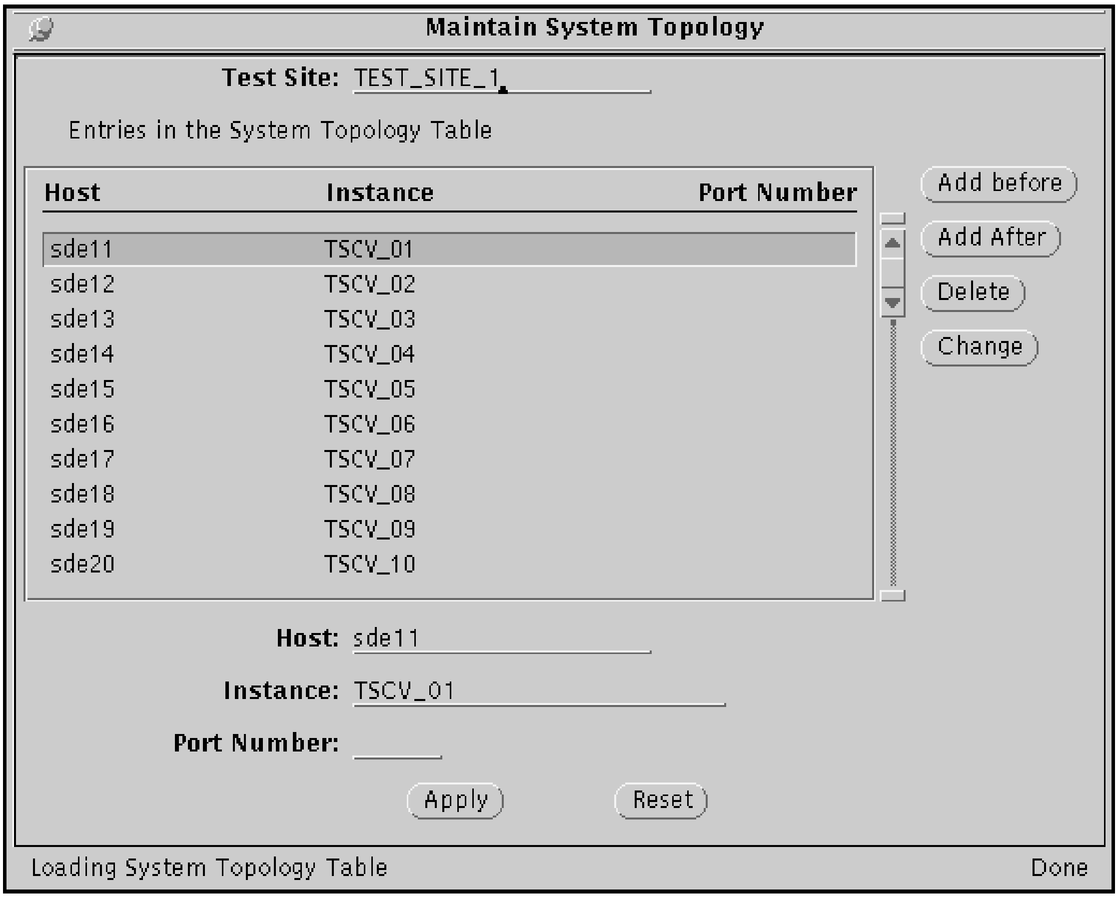


Figure 8–13 : *The Maintain System Topology window*

8.2.2.11.10.2 The ’Test Site’ Text field

The **’Test Site’** text field is an editable text item for display and edit of the test site. Changes are only local until a new *’System Topology Table’* is stored after applying changes.

8.2.2.11.10.3 The System Topology Table List

The **’System Topology Table’** list will be a list of all entries in the *’System Topology Table’* when the *’Maintain System Topology’* window is opened and after **‘Reset‘**. The user may update the content of the list by entering values into the text fields **Host**, **Instance** and **Port Number** (only for SAS) and then request add or change. System Topology Table entries are listed in the same sequence as found in the *’System Topology Table’*, i.e. no sorting will be provided. The columns are:
– Host name
– Instance (type and instance)
– Port number (only applicable for the process type SAS).
No operation is attached to double clicking an entry in the list
Changes are only local until a new *’System Topology Table’* is stored after applying changes, and then the consistency will be checked.

8.2.2.11.10.4 The ’Add Before’ Button

The **’Add Before’** button is used to add a new entry into the displayed list of *’System Topology Table’* entries before the currently selected one. The content of the **’Host’**, **’Instance’** and **’Port Number’** text fields are

validated according to their validation rules. If the content is not accepted, the user is notified and adding a new entry is denied.

8.2.2.11.10.5 The ’Add After’ Button

The **’Add After’** button is used to add a new entry into the displayed list of ’*System Topology Table’* entries after the currently selected one. The content of the **’Host’**, **’Instance’** and **’Port Number’** text fields are validated according to their validation rules. If the content is not accepted, the user is notified and adding a new entry is denied.

8.2.2.11.10.6 The ’Delete’ Button

The **’Delete’** button is used to delete the currently selected entry in the list of *’System Topology Table’* entries. The entry will be removed from the list without user confirmation.

8.2.2.11.10.7 The ’Change’ Button

The **’Change’** button is used to change the currently selected entry in the list of *System Topology Table* entries. The content of the **Host**, **Instance** and **Port Number** text fields are validated according to their validation rules. If the content is not accepted, the user is notified and changing the currently selected entry is denied.

8.2.2.11.10.8 The ’Host’ Text field

The **’Host’** text field is used to display and edit the host name. When selecting an item in the list of *’System Topology Table’* entries, the host of the currently selected item is written into the text field.

The user is free to edit the text field in order to add or change entries in the list of *’System Topology Table’* entries. There will be no checking on the input at the time the list is updated. When selecting **’Apply’**, consistency checks will be performed.

8.2.2.11.10.9 The ’Instance’ Text field

The ’**Instance**’ text field is used to display and edit the process type and instance. When selecting an item in the list of ’*System Topology Table’* entries, the instance of the currently selected item is written into the text field.

The user is free to edit the text field in order to add or change entries in the list of ’*System Topology Table’* entries. There will be no checking on the input at the time the list is updated. When selecting **’Apply’,** consistency checks will be performed.

TSCV will recognize the process type depending on their names. The following list shows how the different applications will be recognized:

HCI_XX – recognized as a HCI application.
TES_XX – recognized as a TES application.
TSCV_XX – recognized as a TSCV application.
TEV_XX – recognized as a TEV application.
CSS_XX – recognized as a CSS application.
DBS_XX – recognized as a DBS application.

other – All other names is assumed to be SAS applications.

where XX must be a number between 01 and 32.

8.2.2.11.10.10 The ’Port Number’ Text field

The **’Port Number’** text field is used to display and edit the port number connected to a SAS. When selecting an item in the list, the port number of the currently selected item is written into the text field. Note that port number is only applicable for process type SAS and when selecting list entries of other process types, the text field will be empty.

The user is free to edit the text field in order to add or change entries in the list of *’System Topology Table’* entries. As the port number only is valid for process type SAS, any editing of the text field for other process types will be omitted when updating the list of *’System Topology Table’* entries.

When requesting to update the list of *’System Topology Table’* entries (add or change) for process type SAS, the port number is checked for validity. A valid port numbers is in the range from 7600 to 7790 in steps of 10, e.g 7681 is invalid but 7680 is valid.
When selecting **’Apply’,** further consistency checks will be performed.

8.2.2.11.10.11 The ’Apply’ Button

The **’Apply’** button is used to update the System Topology Table used by the test system. When pushed, the content of the list of *’System Topology Table’* entries are interpreted and a local system topology table is generated. The consistency is checked (see below). If everything is accepted, a warning dialog gives the user three options:

- **Store and shutdown**
  The test system has to be restarted before the new *’System Topology Table’* will be used. An application shutdown will be performed without any further user confirmation. Active test configurations will be stopped. At last, TSCV is stopped.
- **Store**
  A new *’System Topology Table’* will be stored. Applications must later be restarted before the changes will take effect.
- **Cancel**
  The apply request is cancelled.

The consistency checks performed upon **’Apply’**, are:

- The instances (process type and instance) shall be unique for process types TES, HCI, TSCV, TEV and CSS.
- The SAS names specified for one host, shall be unique.
- Up to 10 DBS instances, all on the same node.
- Only one of HCI, TES, TSCV, TEV, CSS can be hosted on a node, but they can share the same node and also be hosted on the same node as DBS.
- The number of instances is limited to 32 for TES, HCI, TSCV, TEV and CSS.

- The port numbers specified for one host, shall be unique. The limitation rule for port numbers implies that there will be a maximum of 20 SASes per node.

8.2.2.11.10.12 The 'Reset' Button

The **'Reset'** button resets all changes. The content of the *'System Topology Table'* is loaded and displayed. All changes done locally in the *'System Topology Table'* entries, will be lost.

8.2.2.11.11 The Maintain User Profile Window

8.2.2.11.11.1 General

A user profile is the set of information which describes user environment and controls the allowed activities of a CGS user. Like the UNIX /etc/passwd file, it is stored centrally in the file *$CGS_HOME/config/USER_PROFILES.*

The *'Maintain User Profile'* window shows all the current CGS users in a list. It supports adding, changing and deletion of user profiles. Any changes applied on a user profile (pushing the **'Apply'** button in the add/change user profile window or delete a user profile) will immediately be shown in the displayed list and the file (*$CGS_HOME/config/USER_PROFILES*) is updated accordingly.
The list displayed in the *'Maintain User Profile'* window will therefore always display the current status of user profiles.

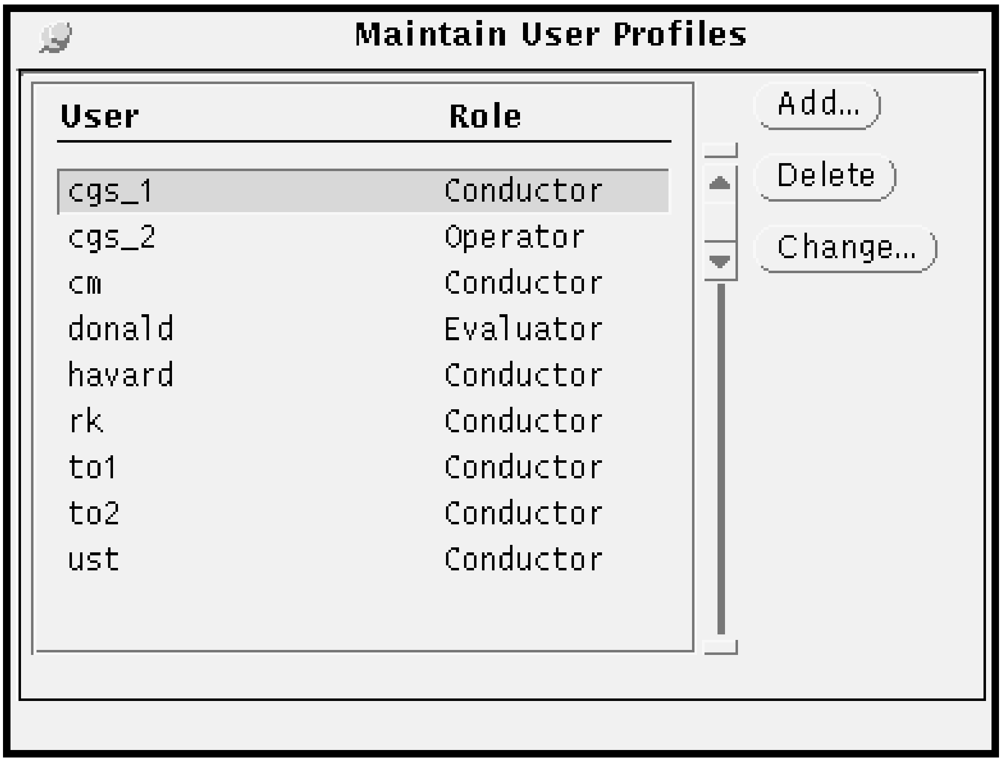


Figure 8–14 : *The Maintain User Profile window*

8.2.2.11.11.2 The User Profiles List

The list will contain all CGS users. The columns of the list are '**User**' and '**Role**'. When double clicking an entry in the list, the *'Change User Profile'* window will be opened. Whenever the *'Change User Profile'*

window is open and selecting an entry in the list, the data associated with the selected entry will be displayed in the *'Change User Profile'* window. Changes done in that window will be overwritten and non–applied changes lost.

8.2.2.11.11.3 The 'Add...' Button

The **'Add...'** button is used to open the *'Add User Profile'* window. The window content is reset ready to specify a new user profile. Note that when editing or adding a user profile and requesting add, the content of the current window (add or change) will be overwritten and non–applied changes lost.

8.2.2.11.11.4 The 'Delete' Button

The **'Delete'** button is used to delete the currently selected entry in the list of user profiles. The user profile will be removed from the list as well as from the file containing user profiles. The user profile will be deleted without any user confirmation.

8.2.2.11.11.5 The 'Change...' Button

The **'Change...'** button is used to open the *'Change User Profile'* window. Attributes of the currently selected entry in the list will be displayed in the *'Change User Profile'* window. Note that when editing or adding a user profile and requesting change, the content of the current window (add or change) will be overwritten and non–applied changes lost.

8.2.2.11.12 The Add/Change User Profile Windows

8.2.2.11.12.1 General

<u>The' *Change User Profile'* window:</u>
The *'Change User Profile'* window is used to change an existing user profile. When the window is open, it will always reflect the user profile of the currently selected user profile in the list of user profiles in the *'Maintain User Profile'* window.

<u>The' *Add User Profile'* window:</u>
The *' Add User Profile'* window is used to specify new users. When the window is opened, it is empty and ready for specification of a new user profile.
Handling of the '*Specific Commands*' will be a bit special. Initially the list of the specific commands will be empty, and the buttons '**Delete**' and '**Change**' will be dimmed. A specification of a specific command can be entered into the text fields. By pushing one of the **'Add Before'** or **'Add After'** buttons, the content of the text field is added as an entry in the list. Then the '**Delete** ' and '**Change**' buttons will be made available.

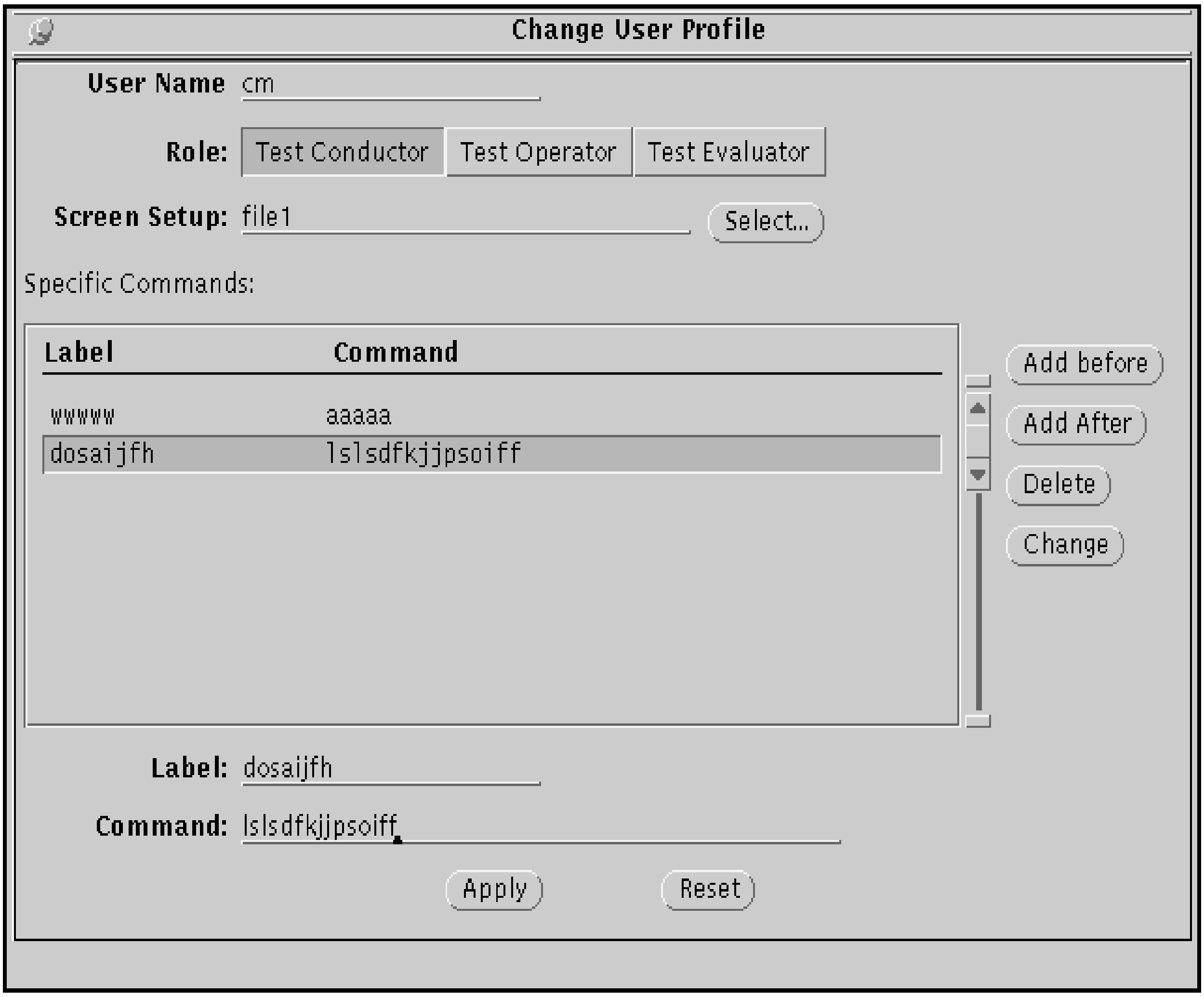


Figure 8–15 : *The Add/Change User Profile window*

8.2.2.11.12.2 The ’User Name’ Text field

<u>The *’Change User Profile’* window:</u>
When the window is opened, the text field contains the name of the user currently selected in the list of users in the *’Maintain User Profile’* window. The text field is not editable.

<u>The’ *Add User Profile’* window:</u>
When the window is opened, the text field is empty. The text field is editable and a new user can be specified. It is mandatory to fill in this text field.

8.2.2.11.12.3 The ’Role’ Choice

The **’Role’** choice will be used to specify the role of the user. The choice is exclusive where the user is granted one of the roles: **’Test Conductor’, ’Test Operator’** or **’Test Evaluator’.**

<u>The *’Change User Profile’* window:</u>
When the window is opened, the **’Role’** choice is set according to the current role of the user.

The *'Add User Profile'* window:
When the window is opened, the **'Role'** choice is set to the default value **'Test Evaluator'**.

8.2.2.11.12.4 The 'Screen Setup' Text field

The **'Screen Setup'** text field is used to identify a file containing the screen setup for the user. All screen setups are centrally stored in the directory *'$HCI_HOME/data/screen_setup_pool'*. Hence, only the file name (not a full path) shall be specified. TSCV will not perform any kind of checking on the text fields, i.e the user is free to specify whatever screen setup he wishes even if the specified name does not exist among the files in the screen setups directory.

8.2.2.11.12.5 The 'Select' Button

The **'Select'** button is used to open the *"Select Screen Setup"* window.

8.2.2.11.12.6 The Specific Commands List

The list contains the specific commands defined, up to a maximum of 8. When an item in the list is selected, the columns **'Label'** and **'Command'** are reflected in the associated text fields. Sorting is not be provided for the list.

8.2.2.11.12.7 The 'Add Before' Button

The **'Add Before'** button is used to add a new entry into the displayed list of *'Specific Commands'*, before the currently selected one. A list item is built from the content of the **'Label'** and **'Command'** text fields. The content of the text fields are not validated.
Note that the change is only local, and has to be applied before the associated user profile is changed.

8.2.2.11.12.8 The 'Add After' Button

The **'Add After'** button is used to add a new entry into the displayed list of *'Specific Commands'*, after the currently selected one. A list item is built from the content of the **'Label'** and **'Command'** text fields. The content of the text fields are not validated.
Note that the change is only local, and has to be applied before the associated user profile is changed.

8.2.2.11.12.9 The 'Delete' Button

When pushing the **'Delete'** button, the currently selected item in the list of *'Specific Command'* will be removed from the list without any user confirmation. The button will be dimmed when the list is empty.
Note that the deletion is only local, and has to be applied before the associated user profile is changed.

8.2.2.11.12.10 The 'Change' Button

When pushing the **'Change'** button, the content of the text fields **'Label'** and **'Command'** is copied into the currently selected item in the list of *'Specific Command'*. The button will be dimmed when the list is empty. Note that the change is only local, and has to be applied before the associated user profile is changed.

8.2.2.11.12.11 The 'Label' Text field

The **'Label'** text field is used to specify *'Specific Command'* The **'Label'** will identify a label to be included in the 'specific commands' menu in HCI. TSCV will not perform any kind of checking on the text field content, i.e the user is free to specify whatever label he wishes

8.2.2.11.12.12 The ’Command’ Text field

The **’Command’** text field is used to specify the full Unix path of a shell or executable (or whatever else ..) to be executed when the user activates the corresponding menu entry in HCI. TSCV will not perform any kind of checking on the text field content, i.e the user is free to specify whatever command he wishes.

8.2.2.11.12.13 The ’Apply’ Button

The **’Apply’** button is used to update the *User Profiles* used by the test system.

The *’Change User Profile’* window:
When pushed, the currently selected user profile will be changed according to the attributes as specified by the *’Change User Profile*’ window. The changes of the user profile will be performed in the list of user profiles as well as in the file used by the test system.

The *’Add User Profile’* window:
When pushed, a new user profile will be added with attributes as specified by the *’Add User Profile*’ window. The new user profile will be added in the list of user profiles as well as in the file used by the test system.

8.2.2.11.12.14 The ’Reset’ Button

The button resets all changes.

The *’Change User Profile’* window:
The content of the *’Change User Profile’* window will be as for the user currently selected in the list of user profiles. Any changed made, will be lost.

The *’Add User Profile*’ window:
The content of the *’Add User Profile’* window will be cleared, and data entered will be lost.

8.2.2.11.13 The Select Screen Setup Window

8.2.2.11.13.1 General

The window is used to select a screen setup to be associated with a user profile.

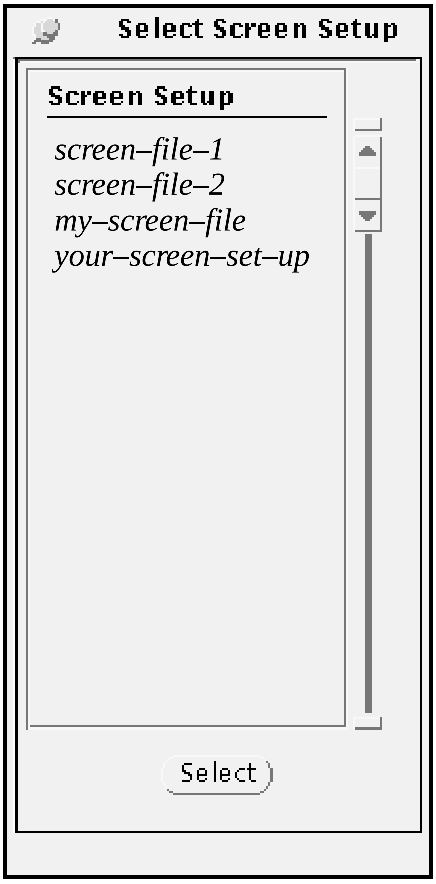


Figure 8–16 : *The Select Screen Setup window*

8.2.2.11.13.2 The Screen Setup List

The list contains all files found at the directory *$HCI_HOME/data/screen_setup_pool*. The files defines screen setups used by HCI. By double clicking at an item in the list, the screen setup is selected and transferred to the associated text item in the *'Add/Change User Profile'* window

8.2.2.11.13.3 The 'Select' Button

When the button is pushed, the currently selected screen setup in the list of screen setups, is transferred to the associated text item in the *'Add/Change User Profile'* window

8.2.2.11.14 The Display Request Window

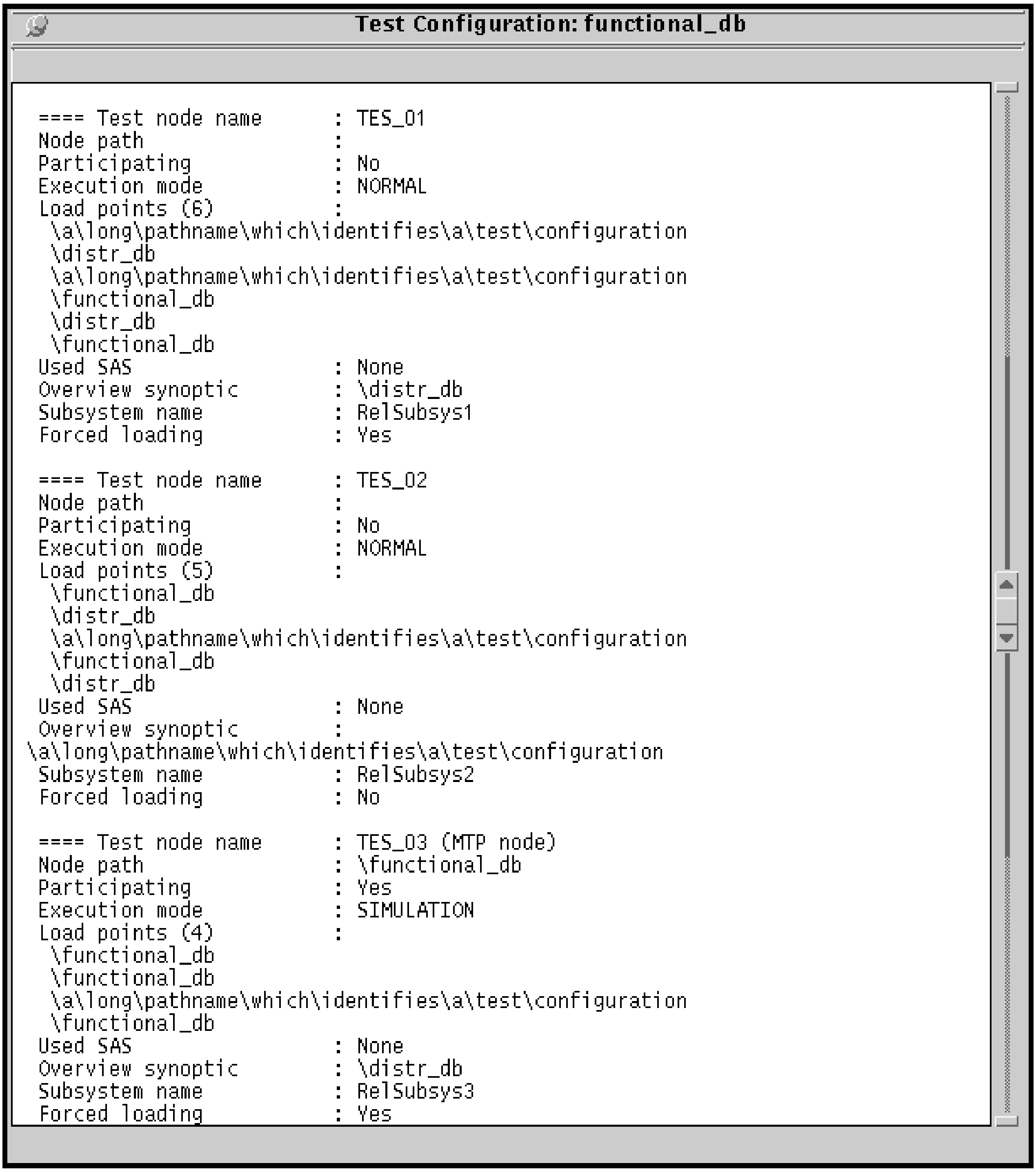


Figure 8–17 : *The Display Request window*

Whenever the user has issued a **View** operation on a selected object, the content of the object will be viewed in the TSCV Display Request Window shown in figure 12. A view operation is either operating a **View** button or to double click in a list. These **View** operations exists:

- **View test configuration** currently selected in the TSCV main window
- **View test configuration** currently selected in the *'Load test configuration'* window.

- **View session key data** in the 'Multi Purpose Test Sessions' window.
- **View software versions** as defined in the version id table.

The window can be closed by pressing the window pin. Each time a 'view' operation is activated, a new Display Request window will be opened. The window has a scrollbar, allowing the user to scroll through the buffer.

The user can not edit the content of the window neither save it on file. A selected part of the window content can be copied, however, and pasted into another window, e.g. as text editor.

Each view request will be displayed in a separate Display Request window. This behavior can be utilized to e.g. compare test configurations.

8.2.2.11.15 Operator Commands and Operations

8.2.2.11.16 TSCV Invocation, Interactive Mode

TSCV has to be invoked as an Unix process by a Unix command. In the integrated CGS, Test System Setup can be selected in a TLUI menu, and TLUI will give the necessary Unix command. It can also be started from the command line of an xterm or command tool window. TSCV will act upon a set of command line parameters.

When invoking TSCV in the interactive mode, it is possible to specify command line options. The command line options are optional.

The command line options that can be used for TSCV, are:

- –geometry {WxH}{[+|–]x[+|–]y}
- –WG {WxH}{[+|–]x[+|–]y} , (short notation for –geometry)
- –position x y , (note that position specified by –geometry has priority).
- –Wp x y, (short form for –position)
- –size w h , (note that size specified by –geometry has priority).
- –Ws w h, (short form for –size)
- –Wi, TSCV comes up open (default behaviour)
- +Wi, TSCV comes up closed (iconified)
- –WP x y , the position if the TSCV icon

An example of the syntax when including the **–geometry** command line option:

start_tscv –geometry +50+75

In fact, also the window size can be specified (although this will hardly be of any use); the syntax is then

start_tscv –geometry 500x400+50+75.

If position is not provided on the command line, TSCV will search the X resource database. It will search for the resource variable **tscv.geometry**, and the variable string should be formatted as **wwwwxhhhhh+xxxx+yyyy**

8.2.2.11.17 TSCV Invocation, Batch Mode

Executing TSCV in batch mode has to be signalled via command line arguments. The following pairs of argument designators and argument values have to be provided, optional arguments in brackets:

–mn <mission_name>
–ec <element_configuration>
–sv <system_tree_version>
–sn <system_tree_node_name>
–cn <CCU_name>
–cv <CCU_version>
–tc <test_configuration>
[–ts <test_session_name> [–fa]]
[–q]

If one or more of the parameters are found to be invalid, TSCV will terminate, providing an exit code identifying the problem.

The set of 'batch parameters' identifying the CCU and test configuration are mandatory:

<mission_name>	is the name of the mission, e.g. "FLIGHT5711"
<element_configuration>	is the name of the element configuration, e.g. "APM"
<system_tree_version>	is the identification of the version of the system name tree to be used for the given mission an element configuration, e.g. "5"
<system_tree_node_name>	is the virtual node in the system (name) tree for the selected element configuration, mission and version of the name tree where the desired CCU is looked up, e.g. "\APM\EGSE"
<CCU_name>	is the name of the CCU in MDB, e.g. "IF_TEST"
<CCU_version>	is the version identification of this CCU. Note that a full version identification comprises the version, issue and revision number, e.g. "1.1.0"
<test_configuration>	is the pathname of the database enditem of type EGSE_TEST_CONFIGURATION. This enditem describes how the setup of the test equipment shall be done, e.g. "\APM\EGSE\SYSTEM\IF_TEST\SETUP_3"

Please note that the MDA **pathnames** on the Unix command line must be **enclosed by single quotes.**

The rest of the parameters are optional.

<test_session_name>	is the name of the test session to be created in the database for the given test,e.g. "IF_TEST_3" (if test_session_name is omitted, all test result data will go into the default test session).

If the **<test_session_name>** parameter is omitted, data produced during the test will go into the 'universal' default test execution session in the database. Note, however, that the default test execution session can only be used by one test configuration.

The optional parameters **fa** and **q** are used as follows:

fa	this argument does not have any meaning unless a test session name is provided. **fa** stands for Final Archive, and if the argument is present, all test result data submitted to the database goes directly into the final archive medium (if the argument is omitted, the test data will only be stored on the database server disk)
q	if this argument is present, TSCV will command the MTP to start the 'Master AP' and then terminate (if the argument is omitted, TSCV will wait for the MTP to signal that the AP has stopped).

After decoding the command line parameters and determining that the batch mode is requested, TSCV will check that the conditions for starting the test are present. Then it will set–up the test system according to the identified test configuration, without any further user interaction. After completing the set–up, it will command the testnode having the MTP role to invoke an Automated Procedure (AP). One of the attributes in the test configuration enditem is MTP_INITIAL_AP, and it is the AP identified by this attribute which TSCV will request TES to invoke.

The MTP_INITIAL_AP must be parameterless!

After starting the AP, TSCV will be waiting for TES to signal that the AP has terminated, unless the optional **–q** option is specified. Then TSCV will terminate immediately after having started the master AP.
When waiting, and TES signals that the AP has terminated, TSCV will also terminate. However, if the optional test session parameter was provided on the command line, it will first close this session.

In the course of checking parameters and checking the start–up conditions, as well as during the set–up and termination, TSCV will send log messages to DBS about the progress and events. If problems are encountered, error messages will also be sent to the CGSI message system.

The Unix process completion code is utilized in the batch mode. Thus, the completion code can be checked to find out if the set–up and test was fulfilled as intended. Completion code 99 represents success. The meaning and implication of most completion codes are rather evident. The event messages from TSCV will aid to trace down the course of the problem more closely.

Two problem situations which may be encountered deserve special description:

1. In the course of the set–up, TSCV will command the MDB software to execute the command **set global default CCU**, where the CCU is the one identified in the command line. TSCV is then obliged to inform the 'participating HCIs' about this event. The set of participating HCI nodes is found from the test configuration enditem. If TSCV fails to deliver the message to any of the participating HCIs, TSCV will produce an error message, however, otherwise ignore it.

2. TSCV will ensure that only one instance of the program is active. If, when started in batch mode, it is found that another instance is already running, TSCV will be terminated with the corresponding error code.

8.2.2.11.18 The Interactive Mode

All error messages created in interactive mode will be displayed to the user in a pop–up dialog window. The messages may categorized as

1 – messages related to the way TSCV is used
2 – messages created due to ill–function of TSCV
3 – messages resulting from error status returned from the other CGS products with which TSCV interacts.

Messages of the first category will in clear text display what the problem is, e.g. that another TSCV instance is already running, or that the user is not authorized to use TSCV.

Messages of the second category represent internal software problems which ideally should have been discovered and removed during implementation and testing. Such errors will be caught in 'exception handlers' in the software, where error messages will be issued. The messages attempts to provide as much information as possible about the condition when the errors occurred. There is little the user can do about such problems, other than report them for maintenance actions.

Messages of the third category attempts to convey to the user the information that the external products provide to TSCV. Simultaneously it is anticipated that the external products logs a message about the problem, and the user should investigate the message log to find out more about the problem and what to do.

8.2.2.11.19 Batch Mode

The final status after completed execution in batch mode, is indicated by the following exit codes:

- Success 99;
- invalid CCU selection 100;
- invalid mission name 101;
- invalid element configuration 102;
- invalid system tree version 103;
- invalid CCU name 104;
- invalid CCU version 105;
- invalid system tree node name 106;
- failed to connect to other CGS product 107;
- invalid test configuration 108;
- failed to setup test configuration 109;
- failed to start test execution 110;
- failed to create test session 111;
- test configuration already active 112;
- node participating in another active test conf. 113;
- test session name already used 114;
- no final archive device available 115;
- failed to start master AP 116;
- another TSCV instance already invoked 117;
- invalid user profile 118;
- invalid command line profile 119;
- default test session already in use 120;
- TSCV internal error 121;
- maximum number of test configurations 122;
- no replay mode allowed 123;

8.3 Test Execution

This section describes how the user having setup the test execution session as described in section 8.1 can perform the test execution using the CGS provided Human Computer Interface (HCI) product called *Online Test Control*.

First section 8.2.1 provides an overview of how the user ’sees’ the test data and commands the unit under test and how the user can interact with that test data during a test execution session.

Section 8.2.2. then describes in detail how the user can execute and monitor the test via the HCI Online Test Control tool.

Finally section 8.2.3 provides some information for advanced users on how the TES behaviour can be controlled by various configuration parameters.

8.3.1 Overview

CGS/VICOS provides a predefined set of ”system display” windows to show the status of the test equipment. It provides dynamic, user controlled display of checkout equipment and unit test data via synoptic displays.

8.3.1.1 Visualisation of Test Data

The following windows and displays are provided to the user :

<u>Synoptic displays:</u>

– provides mimic diagram of the unit under test or test equipment
– contains static part (drawing, graphics, bitmaps)
– contains dynamic output elements dynamically animated with data
– contains user interaction widgets (HLCL commands)
– supports up to eight synoptics per computer screen with 50 dynamic outputs each

<u>Message window:</u>

– displays error or other system messages generated by CGS/VICOS
– type of message can be selected
– source of the message can be selected
– generate audible alarms

<u>Command window</u>

– HLCL commanding of the test equipment or a dedicated test node
– single commands or command sequences
– command sequences from file or from database
– provides command editing capabilites (textedit like)
– provides scrollable command history
– provides possibility to connect to a dedicated test node

<u>Top level menu(”Online Test Control”)</u>

– means to start all HCI windows/applications

<u>Info–bar</u>

– version and time information

System advisory

– overall test equipment system status
– overall status of the unit under test

Tree Explorer

– navigation in name tree
– item specific menus for each name tree object

– out of limit indications in name tree

Test node status window

– detailed status of the test nodes
– acquisition / monitoring / conditions
– stimuli

AP status window

– detailed status of automated procedures running in the test equipment
– current UCL statement
– state of the AP

DB node status window

– detailed status of database node running DBS

SAS status window

– detailed status of the SAS programs running in the test equipment

Graph Tool

– display measurements in different graphical form
– not predefined but online configurable

Raw Data Dump Tool

– display contents of ADUs / Tm packets in raw format
– not predefined but online configurable

8.3.1.2 Commanding the Unit Under Test

CGS/VICOS provides the full range of test control from completely manual to completely automatic using the following:

– HLCL commands and HLCL sequences

– Automated procedures
  – run on the test nodes
  – are compiled and thus run faster and with deterministic timing
  – three priorities: user low, user high, emergency
  – up to 20 AP in parallel on one test node (* 32 test nodes)

– Activate Commands/sequences/APs from Synoptics (via mouse click), from command windows, or as a result of monitoring exceptions.

<u>Use of the High Level Command Language (HLCL)</u>

Interpreted keyboard commands issued from the High Level Command Language (HLCL) provide

- single keyboard command
- sequence of individual keyboard commands

HLCL **sequences** can also be defined by the User (offline in a file) and analysed by the receiving software using the HLCL interpreter. No intermediate compilation is required. The HLCL Interpreter will also access the Mission Database to obtain sequences of HLCL commands.

HLCL allows to call any function of imported **UCL user and system libraries**. In particular, HLCL supports the **invocation of AP**s in any test node, thus establishing a further level of automation in the system as well as interactive access to each test node.

HLCL also allows to **control the CSS** Simulation Function. Several Commands are available to setup the simulation and to set or read simulation parameters.

Refer to Appendix H for a specification of HLCL Commands provided in a Standard CGS Configuration.

HLCL sequence can be executed in a **single step** mode, where the next statement of the sequence is executed only after confirmation by the user. Refer to Appendix H and chapter 8.3.2 of this document.

If defined so in the TES Configuration File, TES keeps the the executed Code for an HLCL statement calling an UCL Library Function in a specific file, as defined in the TES Config. File. This allows for minimum AP debugging support during HLCL Sequence development.

8.3.1.3 System Housekeeping Data

CGS/VICOS provides the following system housekeeping data:
- Mode
- Database load point
- free disc space
- printer status
- Actual time
- etc.

It also provides:
- SAS Status
- AP Status
- Monitoring status
- Stimulus generation status

The values are used in VICOS system displays and may be made accessible via SW Variables to UCL / HLCL.

8.3.1.4 Storing of On–line Data

With CGS/VICOS the user can archive the following:
- raw data packets read from SAS
- data packets (stimuli) sent to SAS
- events needed for replay (starting / stopping SMT)

The archived data are used for test data evaluation and in VICOS replay mode.

CGS/VICOS also allows for logging of the following:
- error situations
- important events (e.g. sending a stimulus, system status changes)
- ”user events”
- engineering values to be be logged on user request

The user may also establish private data in user private files.

All such data is stored in the test result database.

8.3.1.5 Access to Stored On–line Test Data

During on-line test the normal test evaluation tools can be started to examine the data stored in the test result database. Data is really written to test result database ”immediately”. The only prerequisite is that the workstation on which the test evaluation shall occur must have visibility (UNIX, NFS, ORACLE) to the database. Even the same workstation that is used for test execution control can be used for test evaluation.

8.3.1.6 Automatic Data Supervision Features

Monitoring is the automatic supervision of engineering data and the initiation of predefined actions in case of anomalies detected. The data from the unit under test and the test equipment itself can be monitored.

Specific data can be checked against predefined limits such as:
- upper/lower limit
- delta limit check
- danger limit/normal limits
- up to five different normal limit sets with different upper/lower or actions

In case of limit violation the following can take place:
- generate a system error message
- generate a user error message
- issue a command (either to unit under test or checkout equipment)
- issue a sequence of commands
- start emergency automated procedure

For automatic control of data processing functions and applied limit sets **conditions** may be defined in the MDB or online via UCL.

APs written in UCL may be used monitor values and write to software variables, which, in turn are monitored or evaluated by other APs/SAS.

Furthermore, SAS written in Ada may be developped implementing complex data monitoring functions, providing information on out of limit situations as messages or as values written to software variables.

See chapter 8.3.3 for further description.

8.3.1.7 On–line Modifications of the Test Configuration

CGS/VICOS allows for the on–line modification (i.e. by updating the items in the MDB and directly using the updated definitions) of the following;

- automated procedures
- HLCL sequences
- synoptics

VICOS does not allow on–line modification of measurement and stimulus descriptions in the loaded configuration, but the user can change data limits online as well as applied limit sets. Also, some routing information for ADUs and GDU/SWOP_COMMANDS may be overwritten online, and conditions may be created or deleted.

8.3.2 On–line Test Control

The Human Computer Interface (HCI) software provides application/services to control and/or observe the actual running test. In the following the HCI product is called *Online Test Control* because it will not provide the human computer interface to the whole Common Ground Software (CGS), but only to the Test Execution Software (TES).

General aspects and guidelines for the human computer interface (e.g. handling of menus, window layouts) are derived from the HCI Standards based on the OpenLook Styles Guide.

The Online Test Control provides different services and applications to control and monitor the test execution:

- Main Menu
  The main menu starts the Online Test Control applications/services as described below.
- Screen Setup Service
  The screen setup service enables the user to store and load actual screen setups.
- Status Display
  The Status Display displays the overall status of the Online Test Control (like ”Ready”, ”Starting AP Status”, ”Going down”) and the environment in which it is executed (mission data base name, etc.).
- AP Status Display
  The AP Status Display shows the operator a list of the AP’s for one test node provided with status information.
- Online Raw Data Dump
  The Online Raw Data Dump Window enables the human user to make a ’snapshot’ of current raw data and display it in specific format (hexadecimal/decimal) on the screen.
- SAS Status Display
  The SAS Status Display displays the status (name, service announced, number of error messages, last error message, link identifier) of all special application software running on a test node.
- Test Node Status Display
  The Test Node Status Display displays overall status information about a test node.
- Database Node Status Display
  The Database Node Status Display displays overall status information about the database server node.
- System Advisory
  The system advisory shows the operator the overall status of the database server node, the test nodes, and each sub–system tested. These symbols can be opened like icons to show the operator a synoptic picture of the sub–system/node.
- Synoptic Display
  The synoptic display supplies the operator with hierarchical colour synoptic presentation of test data, test object status, test progression status and EGSE status.
- General Graph Facility
  The General Graph Facility provides the operator a general tool to display single measurements and S/W variables in different forms (text, bar chart, etc).

- Online Raw Data Dump
  The Online Raw Data Dump enables the human user to monitor raw data and display it in specific format (hexadecimal/decimal) on the screen.
- Command Facility
  The command facility enables the operator to control and monitor test sessions and the test system by entering HLCL commands.
- Tree Explorer
  The Tree Explorer allows to execute commands via menus defined for each object of the MDB Name Tree. The status and values of measurements may be interrogated as well as commands (TC,GDU) may be sent. Also general commands to control the test system (e.g. enable arcvhiving) may be given.
- AP Input Dialog
  The AP Input Dialogs are provided to prompt for user input during automated procedure execution.

8.3.2.1 Set–up and Initialization

To setup the environment in which Online Test Control may run, the Test Conductor must start the Verification, Integration, and Check Out Software (VICOS) via CGS Top Level User Interface (Task Selector entry ”CGS System Startup”,) and perform the necessary test set–up as described in section 8.1 (Task Selector entry ” Test Setup”).

8.3.2.2 Getting Started

To run Online Test Control power up a workstation that has been configured to a CGS workstation by the EGSE Administrator and login with a user account configured for test execution (see Operation Constraints). In the following examples the user account `cgs_1` is used.

After you have logged in with user name and password, CGS will automatically start the Top Level User Interface under the Open Windows environment.

To start Online Test Control open the *CGS Task Selector* and select *Online Test Control* from the *Select Task* menu as shown in Figure 8–18.

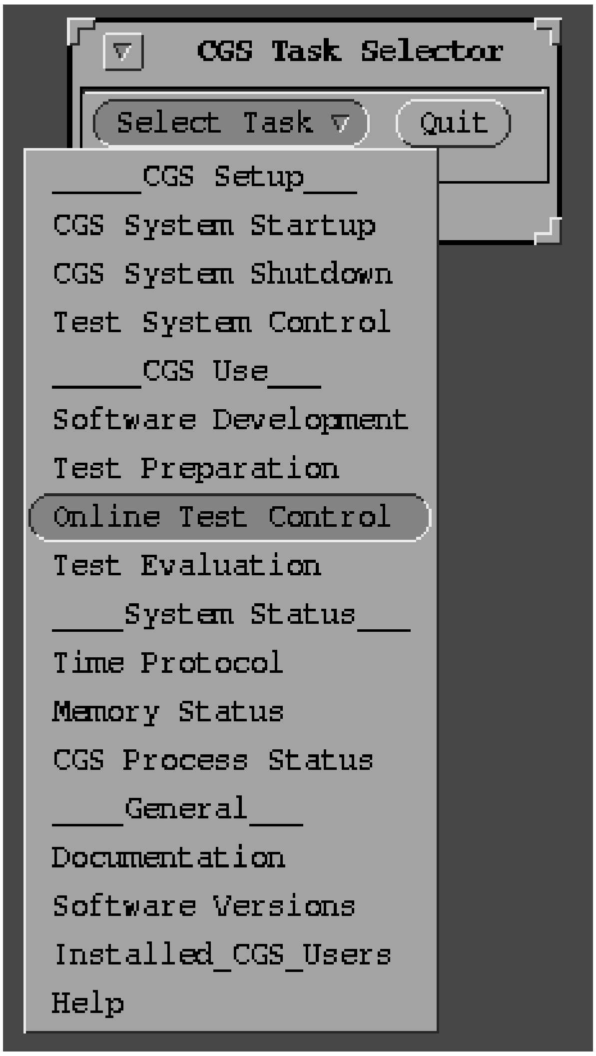


Figure 8–18 : *CGS Task Selector*

After a few seconds, Online Test Control will come up with its status display and main menu (it is not indicated by the CGS Task Selector that invoking Online Test Control is still in process, so be patient). When going into operation, Online Test Control displays its main menu in the upper left corner and the status display at the bottom of the screen (this may vary depending on the default screen setup defined in the user profile).

When the status display is shown you may follow the progress of Online Test Control going in operation.

When Online Test Control shows that it is ready, the main menu is enabled and all functions are available. You can, for example, start the Clock application by selecting *Online Test Control* –> *Data Displays* –> *Clock* –> *Normal* from the Online Test Control menu as shown in Figure 8–19.

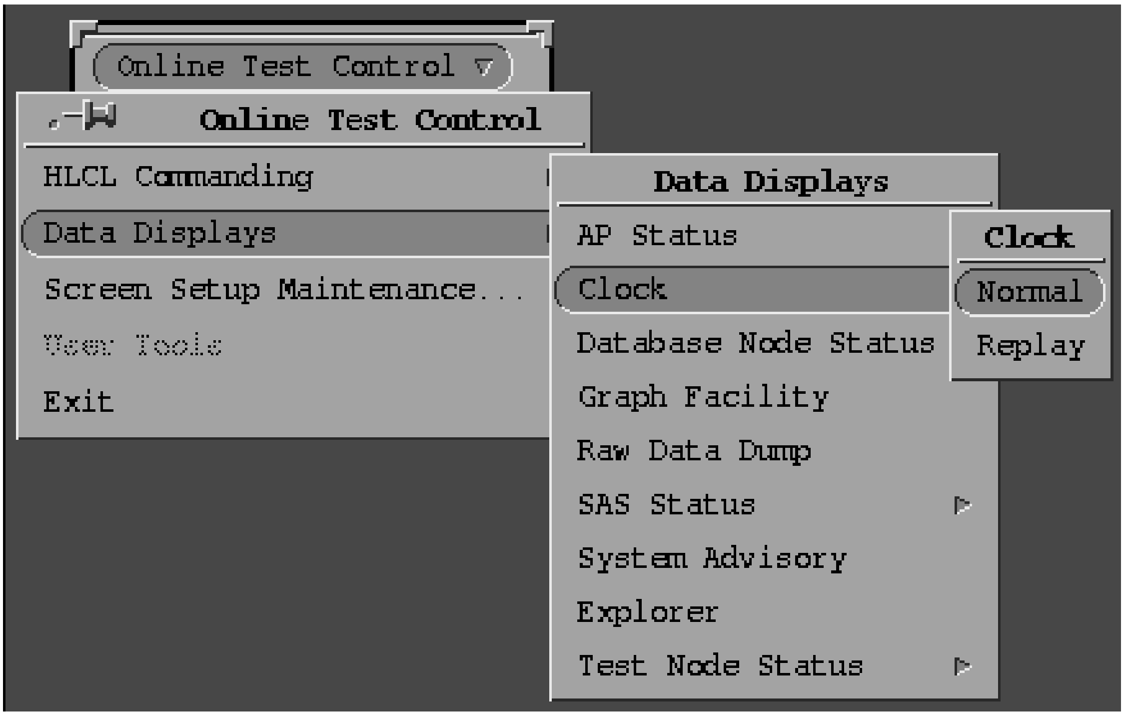


Figure 8–19 : *Start Clock Application*

8.3.2.3 Normal Operations

After getting started (see section 8.3.2.2) Online Test Control comes up with at least the Online Test Control Menu and the Status Display.

The Status Display (see Figure 8–20) shows the status about the Online Test Control software. It displays the node name or HCI Identifier of the Online Test Control, its software version, the actual Configuration Control Unit (CCU) with the version number, and the overall status. The Synoptic Display Control Panel is described in section 8.3.2.3.2.13, Synoptic Display.

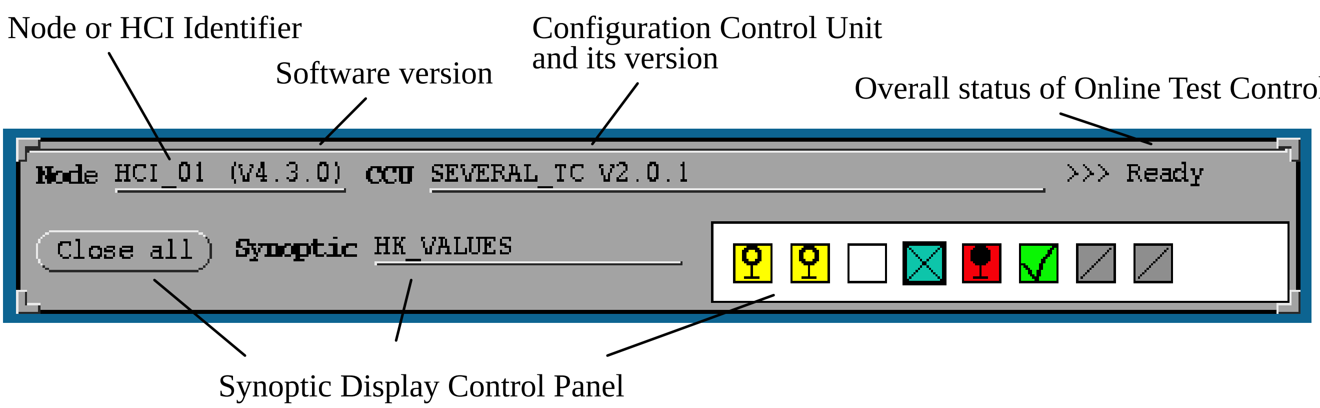


Figure 8–20 : *Status Display*

When going into operation the Status Display shows the following progress messages:

>>> Going into operation. Be patient!

>>> Loading user profile...

>>> Starting communication services...

>>> Connecting to test result data base...

>>> Opening mission data base...

>>> Initializing data views...

>>> Validating data views...

>>> Loading test configuration...

>>> Selecting data base configuration...

>>> Loading error messages...

>>> Connecting to \A_PATHNAME\MTP

>>> Starting MAIN_MENU

>>> Starting STATUS_DISPLAY

>>> Ready

The status messages are shown for only a few milliseconds to some seconds (especially the operations connecting to test nodes may take some seconds if the test nodes aren’t available).

When ready the Online Test Control Menu is enabled and all applications and services are selectable. Figure 8–21 shows the Online Test Control Menu pinned up after selection of the pin button. When the menu is pinned up it can be moved on the screen, but is still available by selecting the *Online Test Control* button.

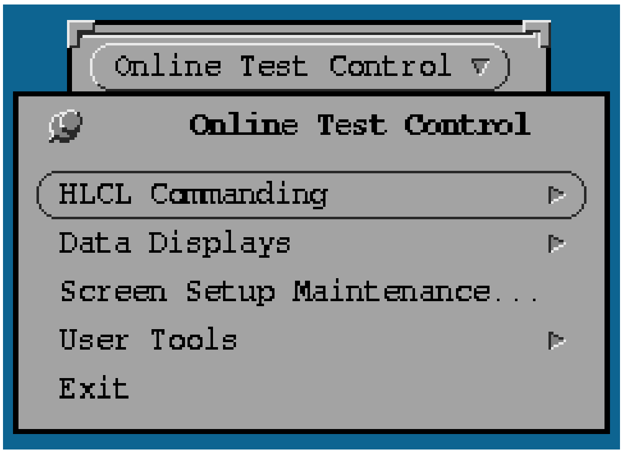


Figure 8–21 : *Online Test Control Menu (pinned up)*

The following sub–sections list all applications and functions provided by Online Test Control. The sub–sections are structured according to the Online Test Control Menu.

8.3.2.3.1 HLCL Commanding

When selecting *HLCL Commanding* from the *Online Test Control* menu the HLCL Command Facility will be started with the Master Test Processor set as default node, i.e. all remote commands (e.g. UCL library routines like suspend_ap) will be routed to the Master Test Processor.

To choose another test node as default, pull to the right to pop up the Test Nodes submenu. It provides a list of all test nodes defined in the test configuration. If a test node is not available (e.g. has not been started) the menu item is not selectable and the label is displayed in gray.

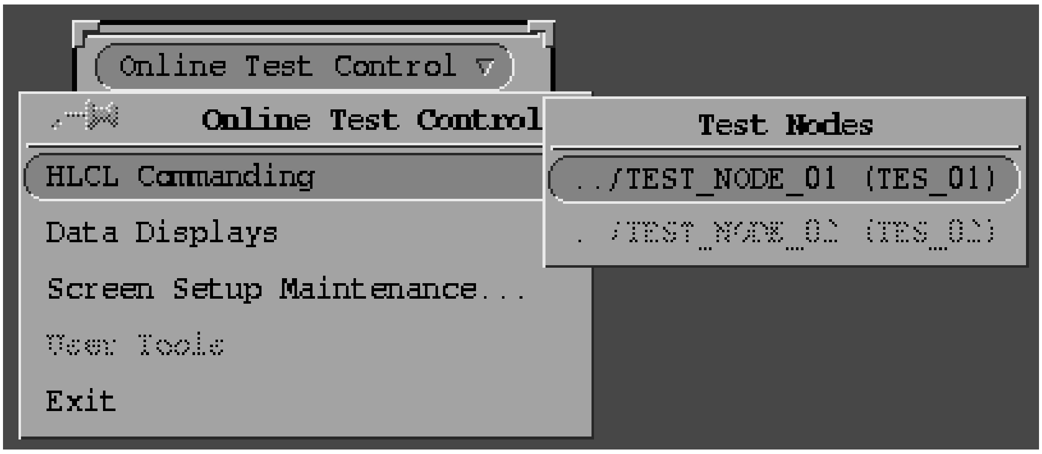


Figure 8–22 : *Test Nodes Submenu*

Invoking an HLCL command tool

- Select ***Online_Test_Control –> HLCL Commanding***

The command window appears on the screen. It consists of an input area and a scrollbar (see Figure 8–23). The command window is ready to accept command input if it shows a new prompt ”HLCL:” with the text cursor symbol behind it.

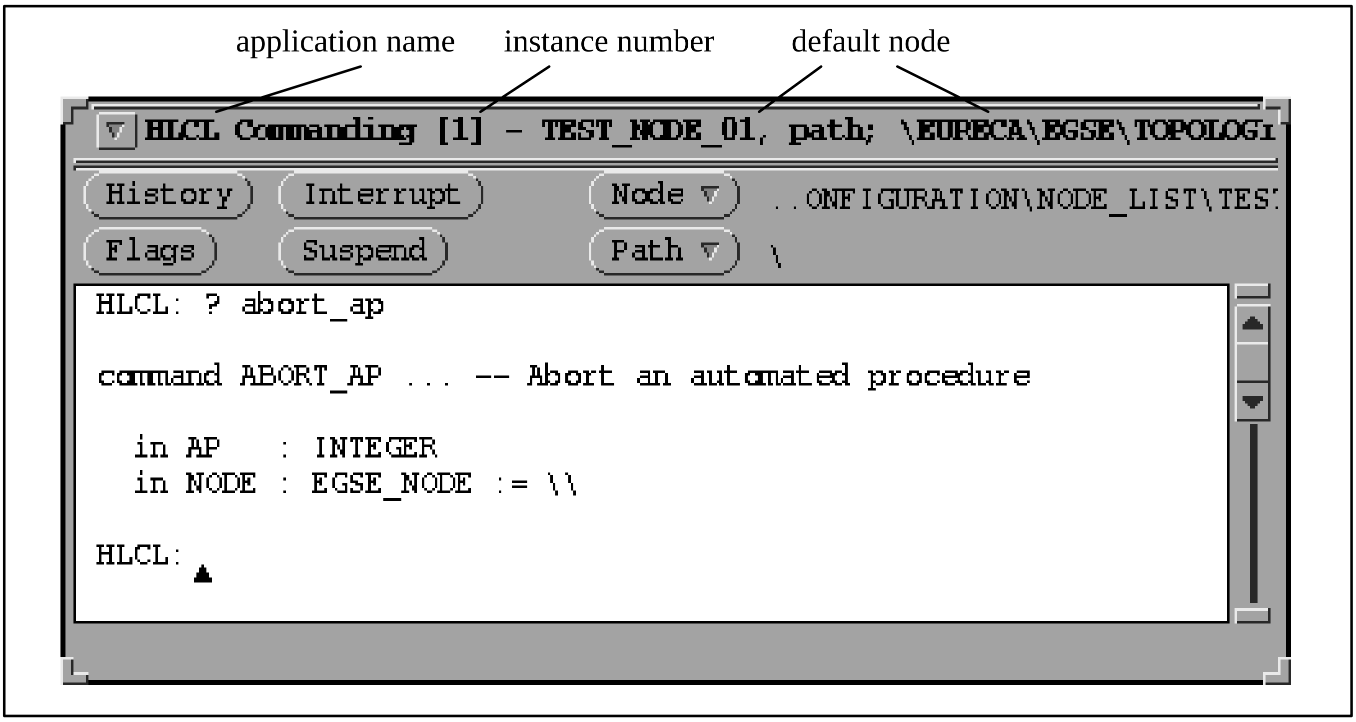


Figure 8–23 : *HLCL Commanding*

To support the command input the following text edit functions (see below) are provided:

- Text insertion, word wrapping, text deletion, and pasting (but only for the actual command)
- Selection and copying (everywhere in the input area).

A command input is terminated by pressing the <Return> or <Enter> key, that will cause the execution of the command.

To get help enter a question mark (<?> key) followed by the <Return> key.

Command Editing

Use text–editing keyboard and menu commands to write and change plain ASCII text.

To set the editing caret, position the mouse pointer and click SELECT. To move the caret, use the arrow keys on the right keypad.

To select text, click SELECT at the beginning of the text, then click ADJUST at the end.

You can copy or cut the selected text using the **Copy** and **Cut** keys on the left keypad. With the Paste key, you can then **paste** the text elsewhere in this or another text window.

The standard keys for **find, again** and **undo** are also available. **Delete** or **Back Space** are deleting the character left to the cursor.

Copy, Cut, and Paste are available on the Command Window Edit menu, along with many other options. To display that menu, click MENU in the input area.

Additional to text edit functions the command window provides a command history function to recall previous entered commands.

Command History

- Pressing the <cursor up> arrow retrieves the predecessor of the displayed command.
- Pressing the <cursor down> retrieves the successor of the displayed command.
- To clear the command line press the ESC–key or CTRL C
- or **Click** with the **left mouse button** on the ***Interrupt*** button.
- **Click** with the **left mouse button** on the ***History*** button. A command history window appears as shown in Figure 8–24.

Note that in contrast to the history window the history buffer (retrieval with cursor keys) has a fixed size.

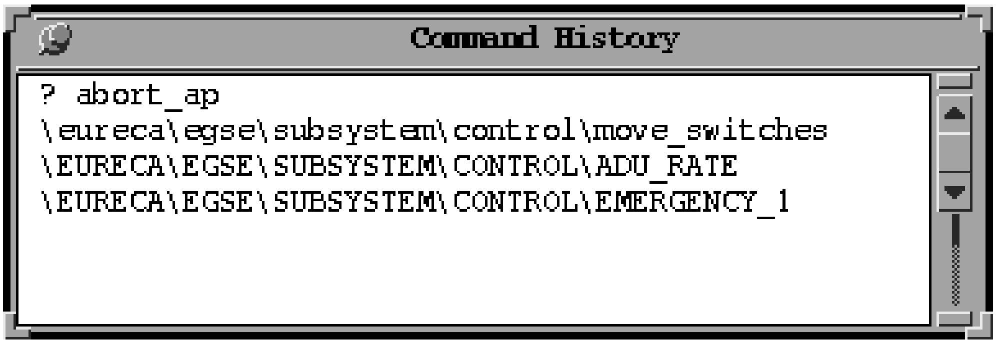


Figure 8–24 : *History window*

The cursor key history actions will delete the text after the prompt in the command window and display the command from the history instead of it. If the top (bottom) of the history buffer is reached no command is displayed. To reenter the command list press the <cursor down> (<cursor up>) key.
It is possible to edit the ”recalled” commands afterwards.

The user may select one or several commands from the history window and copy/paste them into the command window to execute them.

To cancel the execution of a command (e.g. HELP) the user has to following possibilities.

Command Cancelling

- Press the ESC–key or CTRL C.
- Or **Click** with the **left mouse button** on the ***Interrupt*** button.

The cancel of the command execution is confirmed by ”Command cancelled” and a return to the prompt.

Note that only some commands can be cancelled.

To stop the execution of a command sequence the user has to following possibilities.

Suspending a command sequence (with implicit cancelling of the currently executing command**)**

- Press the ESC–key or CTRL C.
- Or **Click** with the **left mouse button** on the ***Interrupt*** button.

Suspending a command sequence (without implicit cancelling of the currently executing command**)**

- **Click** with the **left mouse button** on the ***Suspend*** button.

If a command sequence is currently suspended the *Suspend* button will change into a *Resume* button.

Note that only one command sequence can be suspended at a time. If a new sequence is executed and suspended there is no more a resume possibility for the previously suspended sequence.

Resuming a command sequence

- Enter the HLCL command ”Resume”.
- Or **Click** with the **left mouse button** on the ***Resume*** button.

A confirmation request is given for commands which are specified for ”confirmation”.

Command Confirmation

- To execute the command type ”y” or ”Y”, to cancel the command type ”n” or ”N” or simply press <Enter> because the latter is the default.

If the user enters an incomplete command the interpreter will ask for missing mandatory parameters.The missing information is prompted for in a loop until the information could be accepted without errors.

Entering missing parameters

- Enter the requested parameter end press <Enter> or <Return>.

To terminate the interpreter’s asking simply

- Press the ESC–key or CTRL C
- or **Click** with the **left mouse button** on the ***Interrupt*** button.

A default node or a default path can be chosen in the following way.

Setting a default node

- Enter the HLCL command ”Default_Node := ...”.

If the node has been a default node before it is possible to

- **press** the ***Node*** button with the **right mouse button** and select a node as shown in .

Note that the command window only keeps track on the last four default nodes.
The <no node> item (\\) is always present in addition to the maximum of four nodes.

Setting a default path

- Enter the HLCL command ”Default_Path := ...”

If the default path has been a default path before it is possible to

- **press** the ***Path*** button with the **right mouse button** and select a path.

Note that the command window only keeps track on the last four default paths. The <no path> item (\) is always present in addition to the maximum of four paths.

The flag window is one that eases the setting of some global variables

Invoking the flags window

- **Press** the ***Flags*** button with the **right mouse button**. The flags window appears on the screen as shown in Figure 8–25.

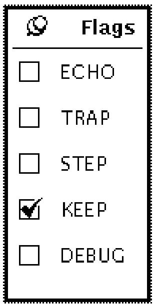


Figure 8–25 : *Flags window*

The flags can be set by clicking on the check boxes. They are used to control command sequence execution and the logging of commands into special command log files.

For more information about HLCL commanding refer to Appendix H of this document.

HLCL Login Sequence

Whenever a HLCL Interpreter is created either by creating a command facility or by a synoptic display, Online Test Control searches for a file called hlcl_login.seq in the user's directory $HOME/.user. This HLCL sequence file is then executed immediately after creation of a command facility or synoptic display. It's purpose is to execute some general HLCL command like alias definitions or library imports.

The location of the HLCL login sequence may be configured by setting some attributes in the Online Test Control configuration file, hci.ini. The configuration file includes one group, called HLCL, providing attributes to modify the location of the HLCL login sequence. The entry point to the login sequence file is defined by the attribute SequenceDirectoryVariable, its default is HOME, i.e. the login sequence is located in the $HOME directory. The attribute LoginSequenceFile specifies the rest of the login sequence file name. This can be a simple name like my_hlcl_login or a name including subdirectory names like the default value (/.user/hlcl_login.seq).

Example: The CGS users shall share the same login sequence (e.g. $GSAF_HOME/cgs/data/login.hlcl) instead of using an individual one.

1) Open hci.ini with an editor as CGS Administrator, e.g. > vi $HCI_HOME/config/hci.ini

2) Search for the group [HLCL] or create it if not defined.

3) Add the following attributes

[HLCL]
SequenceDirectoryVariable = ”GSAF_HOME”
LoginSequenceFile = ”/cgs/data/login.hlcl”

4) Store hci.ini and start Online Test Control (entry Test Execution of the Task Selector).

Configuration

The length of history can be modified in the [COMMAND_FACILITY] section of hci.ini (see also 8.3.2.5), default is 200 commands.

```
[ COMMAND_FACILITY ]
History = 200
```

8.3.2.3.2 Data Displays

To call the Data Displays submenu, select
Online_Test_Control –> Data Displays

The menu then allows to call one of the services as shown in Figure 8–26.

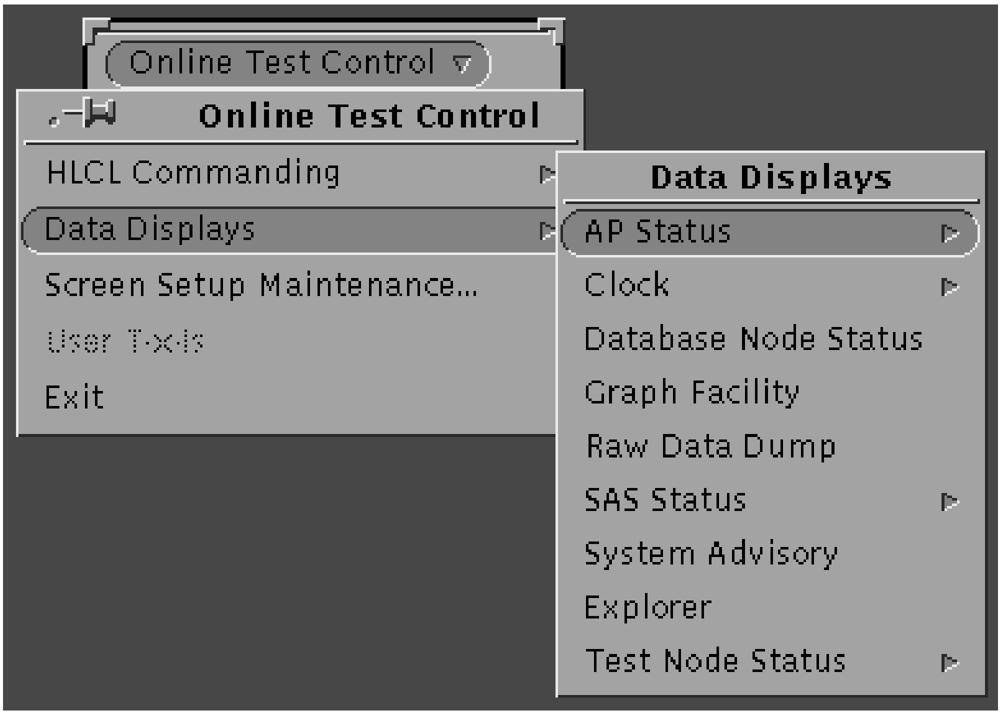


Figure 8–26 : *Data Displays Submenu*

8.3.2.3.2.1 AP Status

To call a AP Status window, select

Online_Test_Control –> Data Displays –> AP Status

The AP Status window (see Figure 8–28) shows an overview of the currently executed APs of one specific test node. The test node can be selected using the Test Nodes Submenu (see Figure 8–27). If no test node was selected the master test processor is used by default. The name of the test node is displayed on the window header. The AP information is displayed in a scrollable list. The APs are sorted according to their identifier (Id). The number of APs displayed are dependent on the vertical size of the window and are adopted when the window is resized. If the window size is modified in horizontal direction the AP Name is truncated or expanded beginning at the front of the name.

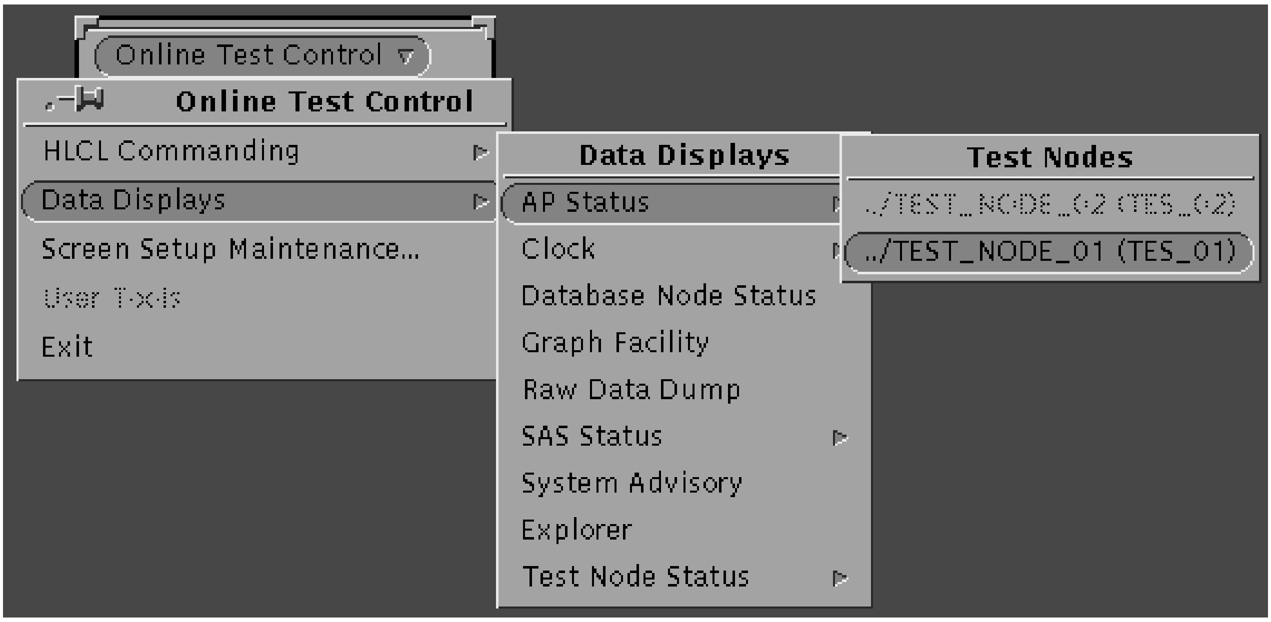


Figure 8–27 : *Test Node Selection*

The following items are shown on the AP list:

- AP_Name — Pathname of the Automated Procedure. If the pathname is too long to fit on the name field it is truncated on the left side. This is indicated by leading dots.
- Status — Actual Status of the AP

NOT_RUN	AP is not running
INITIAL	AP is loaded, but not executed yet
RUNNING	AP is executed
SUSPEND	AP is suspended
TERMINATE	AP is terminated and removed from execution

- Id — The identifier (number) assigned by TES to each active AP
- HCI_Id
  The Online Test Control (HCI) identification from which the AP or parent AP was started.

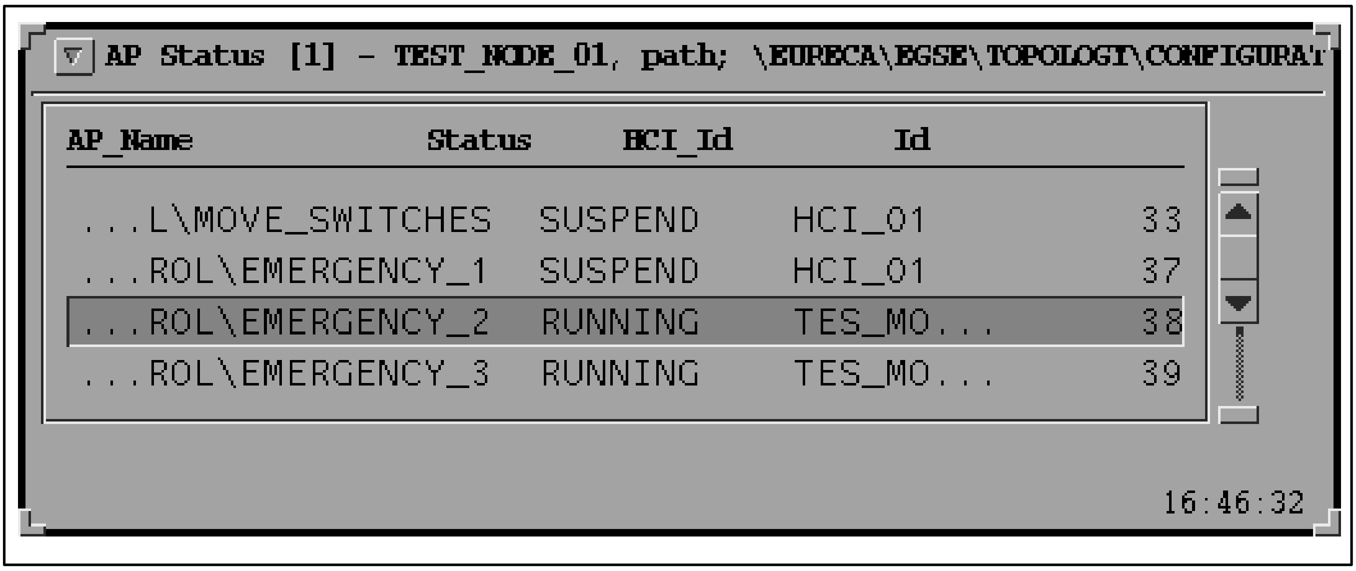


Figure 8–28 : *AP Status*

The AP Status window is updated every 5 seconds. The time stamp of the status data is displayed on the left footer.

Abbreviated information can be made visible by selecting the corresponding item on the AP status list, e.g. selecting the second item with Id 104 will pop up the complete information on the AP name as shown in Figure 8–29. The complete information box pops up at the bottom of the scrolling list and can be removed by selecting the pin.

The complete information window displays an additional field, the UCL statement counter:

- Stmt Number of UCL statement currently executed

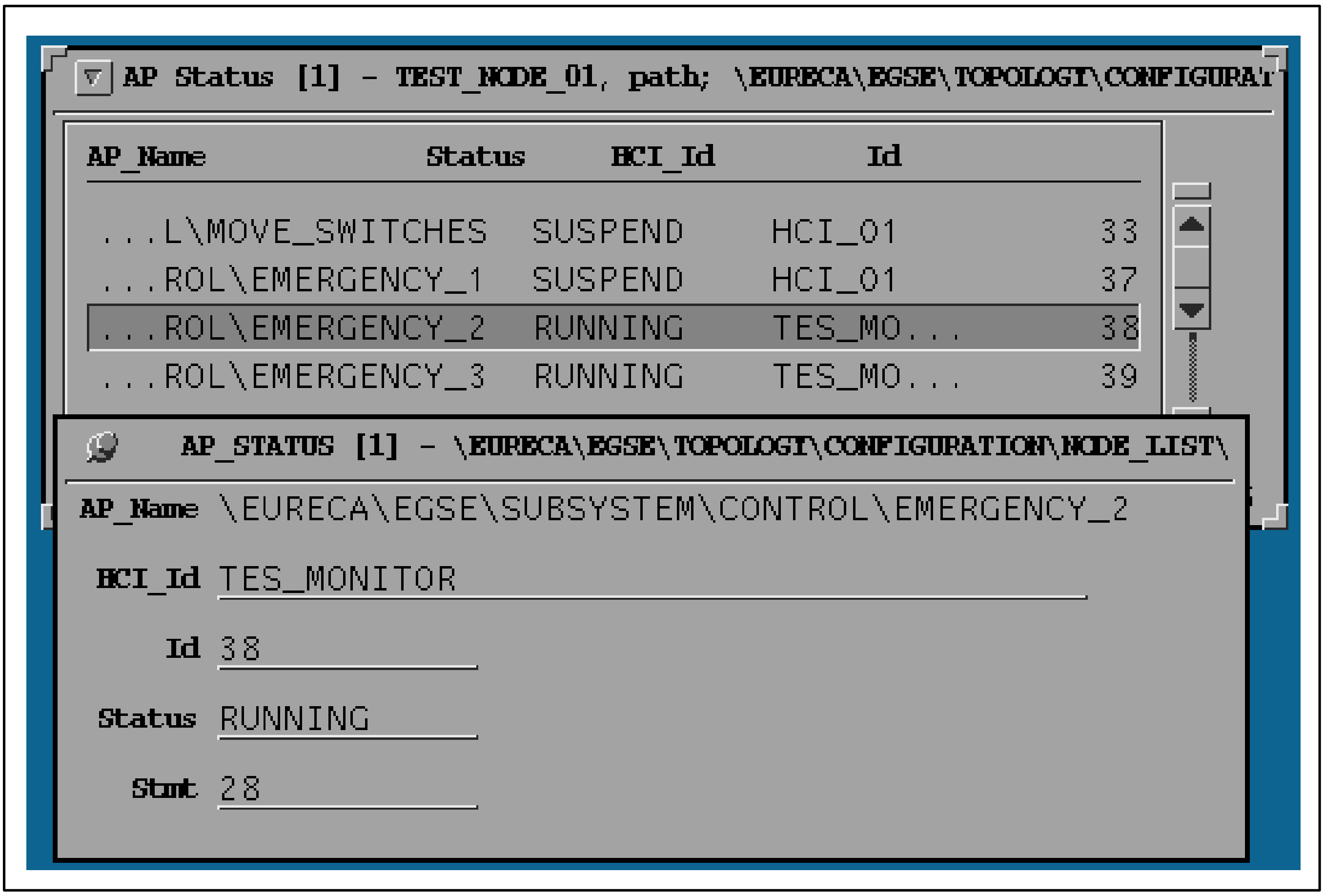


Figure 8–29 : *AP Status Complete Information*

Configuration

The update rate of the AP status window can be modified in the AP_STATUS section of hci.ini (see also 8.3.2.5). To get the AP status window updated every 10 seconds, set the UpdateRate to 10. If the UpdateRate is set to 0, the AP Status window will be updated only if one or more of the values have been changed.

```
[ AP_STATUS ]
UpdateRate = 10
```

8.3.2.3.2.2 Clock

To call the Clock, select

Online_Test_Control –> Data Displays –> Clock

The clock window can be started in two modes, normal and replay mode. In normal mode, the clock window shows the actual time and date of the local time and the simulated mission time (see Figure 8–30). The time is displayed in hh:mm:ss format, the indicator LT is given for local time, the indicator SMT for simulated mission time. The date is displayed in dd.mm.yyyy format. If a time is not available only the indicators are displayed. Figure 8–31 shows a clock that can’t display the simulated mission time.

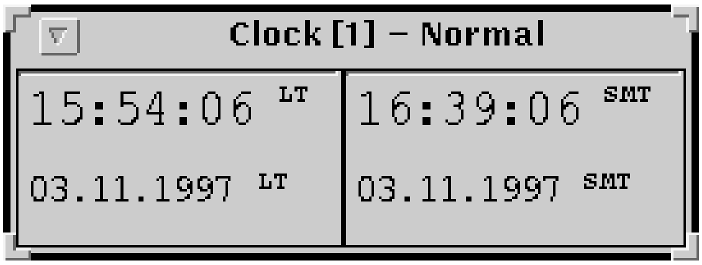


Figure 8–30 : *Clock*

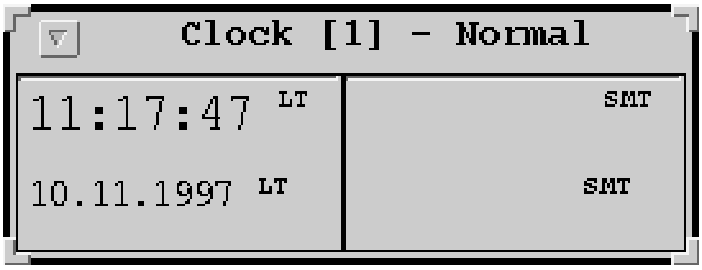


Figure 8–31 : *Clock (SMT not available)*

In replay mode, the clock window displays the recorded local time and simulated mission time as read from archived data by the master test processor. This clock window has an optional footer (see configuration) to show error stati.

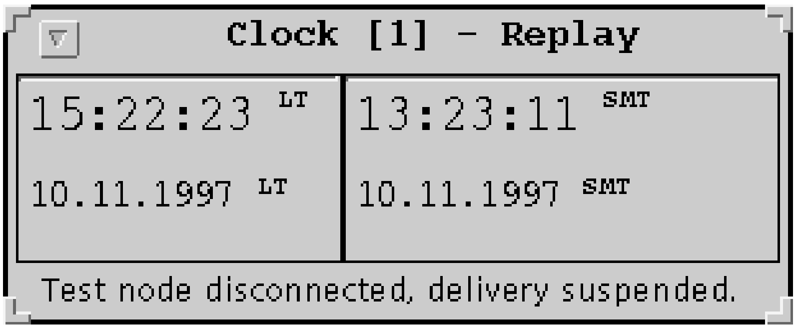


Figure 8–32 : *Clock in Replay Mode*

Configuration

To disable the footer of the clocket the attribute ReplayFooter of the CLOCK group to false (see also 8.3.2.5).

```
[ CLOCK ]
ReplayFooter = no
```

8.3.2.3.2.3 Database Node Status

To call the Database Node Status window, select
Online_Test_Control –> Data Displays –> Database Node Status

The Database Node Status window (see Figure 8–33) displays status information about the CGS Database Server Node. The status data are only available if the Master Test Processor is up and at least in idle mode.

The following status information is provided:

- Printer Status
  The status of printer 1 and printer 2
  Values are ENABLED, DISABLED, and OTHERS
- Printer Queue Status
  The status of print queue 1 and print queue 2.
  Values are OFF, READY, PRINTING, NO_PAPER, NO_TONER
- Printer Jobs
  The number of jobs in print queue 1 and print queue 2
- Session Name
  The name of the actual test session.
- Magnetic Disc Free Space
  Free disc space in bytes.
- Evaluations connected to TRDB
  The number of evaluation users connected to the test result database.
- DBS Overall Status
  The overall status of the database server node.
  Values are OK, NOT_OK, OTHERS.
- TRDB Table Space
  Space (in percent) occupied within the different test result database tables.

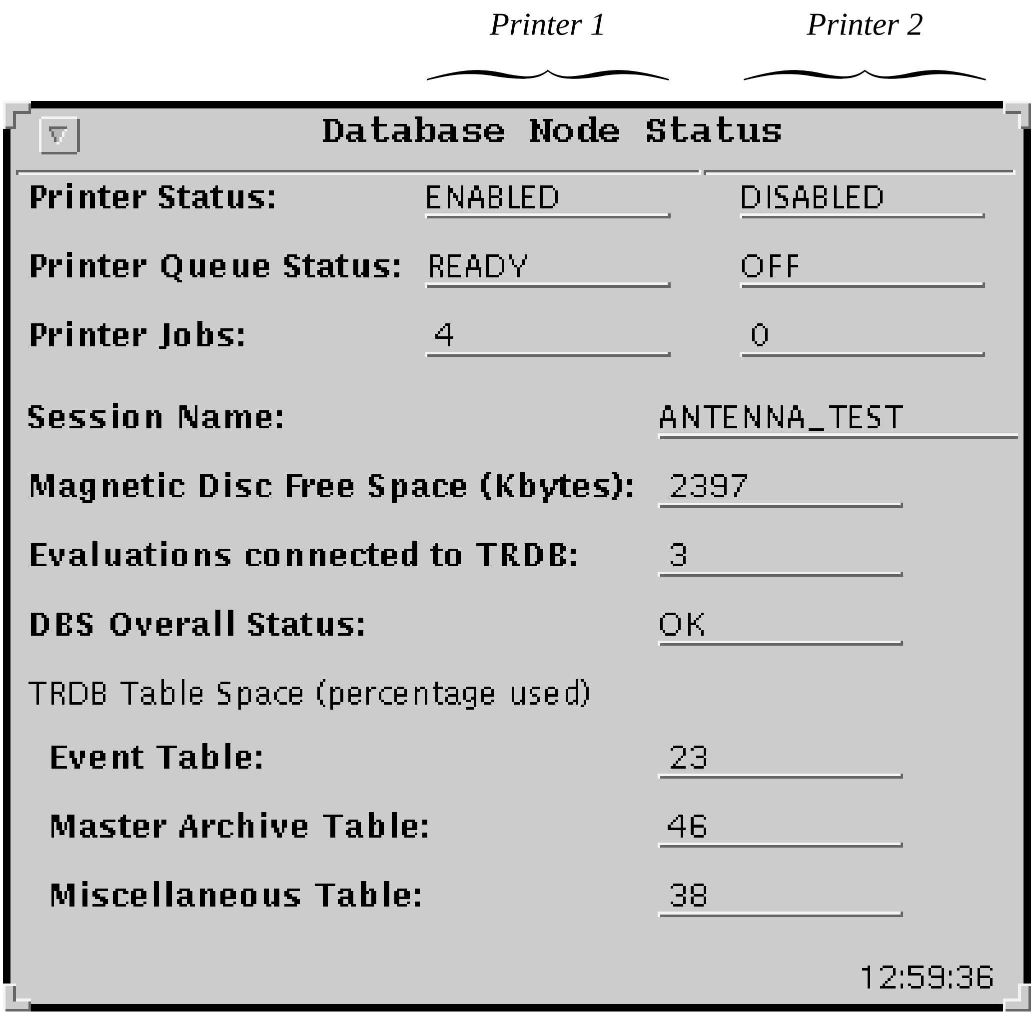


Figure 8–33 : *Database Node Status*

8.3.2.3.2.4 Go/Nogo Window

To call the Go/Nogo Window, select
Online_Test_Control –> Data Displays –> GONOGO Window

The Go/Nogo Window shows the end items of a specific test node that are actually out of limit (NOGO). The test node can be selected using the Test Nodes submenu (see section AP Status 8.3.2.3.2.1). The maximum number of items displayed in rows is 200.

The total number of items that are out of limit is displayed in a text field labeled ’Items Out Of Limit’. Next to this field is the Clear All button. Pressing this button invokes the deletion of all items from the window, so it is like starting the window again.

The following information of an end item subdivided into 17 columns are available for the user:

1	Enditem Name	
2	Enditem Description	
3	NOGO Time	(the time when the item status turns to NOGO)
4	NOGO Engineering Value	
5	NOGO Raw Value	
6	Actual Engineering Value	
7	Actual Raw Value	
8	Actual Limit Set	
9	Nominal Limit Low	
10	Nominal Limit High	
11	Danger Limit Low	
12	Danger Limit High	
13	Acquisition Status	
14	Processing Status	
15	Monitoring Status	
16	Delta Monitoring Status	
17	Pathname	(the complete pathname of the end item)

Save and load of item configurations

If you have a configuration (order and length of columns) you want to load later again, select *Confuguration –> Save* and then the Save File Chooser Dialog appears. Go to the folder where you want to save the configuration. Write a filename you want to use to the Save text field and press the Save button. If no error message is reported the configuration is saved. All filenames of Gonogo WIndow Configurations have the postfix “.gon” .

If you want to load a saved configuration go to the folder where the file is saved, select the file name in the list with a click and then press the Open button. If it is not a valid configuration file a error message is reported else the actual Gonogo Window list is deleted (!) and the new configuration is loaded.

See section 8.3.2.3.2.5 (Monitoring Window) for handling of the file chooser.

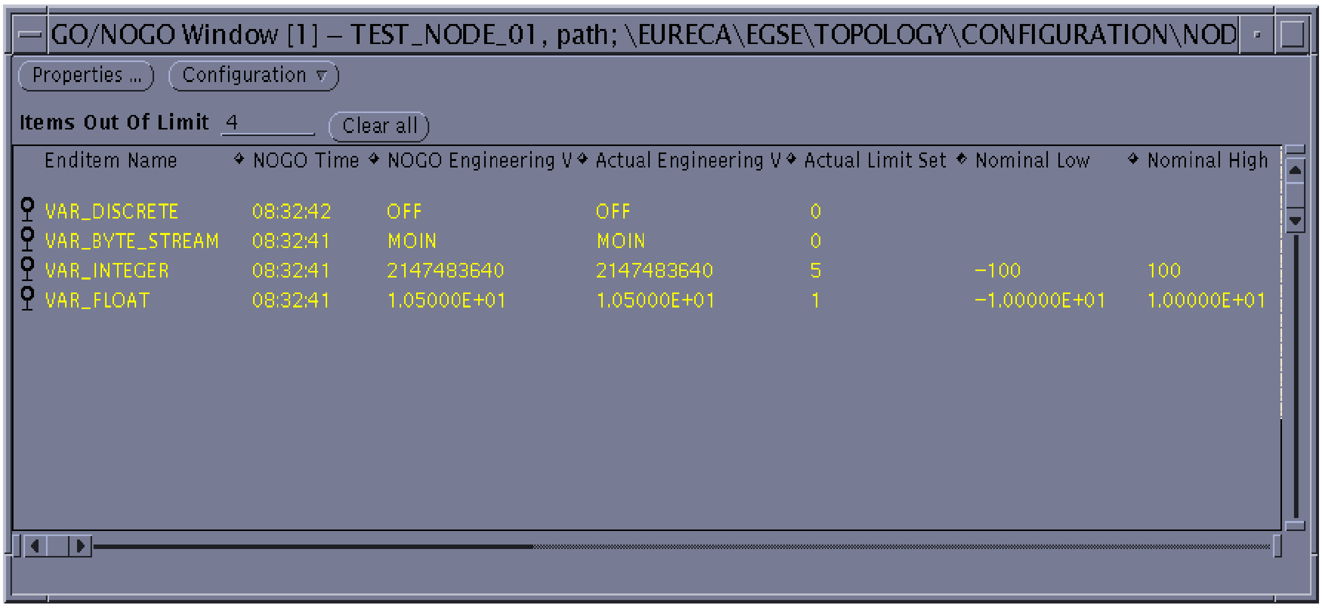


Figure 8–34 : *Go/Nogo Window*

Properties

The size of all columns can be changed: Select the separator ◆ at the end of a column with the left mouse button, keep the button pressed and then drag the separator to a new horizontal position. After releasing the button the column have a new size.

Pressing the Properties button opens the properties window. The column names and their position are shown here. The order of the columns can be changed: Press the button next to the column name with the right mouse button. A menue appears where a new column position can be selected or the column can be disabled, so that it is not displayed anymore in the Go/Nogo Window.

The raw values can be displayed as binary, decimal or hexadecimal numbers.

Additionally the gaps between columns and rows can be scaled up and down.

Press the Apply button to accept all changes that are made or press the Cancel button to reject them.

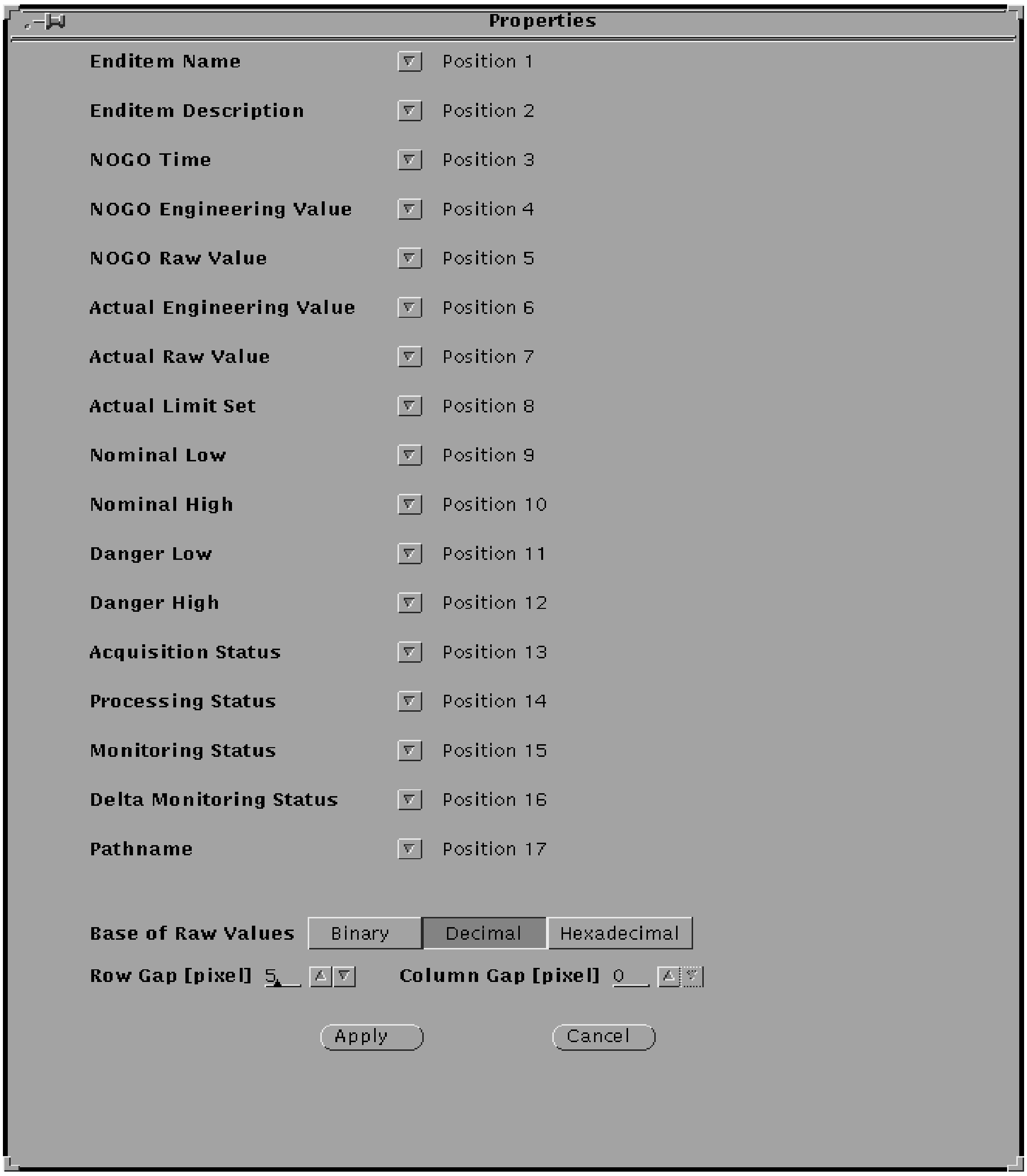


Figure 8–35 : *Go/Nogo Window Properties*

Special Actions

There are special actions defined on displayed items:

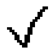
- remove temporary
  Removes this item from the window until its status changes from GO (in limits) to NOGO (out of limit) again.
- remove forever
  Removes this item from the window but it is not displayed again whatever status it might have.
- keep
  Keeps diplaying this item whatever status it has.
- automatic
  The item will be automatically removed from the window if its status changes from NOGO to GO. It is the default behaviour of all new items in the Go/Nogo Window.

To perform one of these actions on an item press the left mouse button on the item row to select it. A border around the item row appears. Then press the right mouse button on the selected item row. A menue pops up where you can select an action you want to execute.

If you press the Clear All button all items are ’automatic’ again.

Colors and Symbols

According to the actual monitoring and delta monitoring status of the end item the row is signed with a special symbol and painted in a special color defined in the hci.ini file:

(Delta) Monitoring Status	Symbol	Color (default)
DISABLED	no symbol	black
IN_LIMITS		green
SOFT_LIMIT_VIOLATION		yellow
DANGER_LIMIT_VIOLATION		red

If the window is iconized the background color of the icon is painted in

- red
  if one displayed item is in DANGER_LIMIT_VIOLATION
- yellow
  if one displayed item is in SOFT_LIMIT_VIOLATION and none in DANGER_LIMIT_VIOLATION
- green
  if one displayed item is IN_LIMITS and none in DANGER_LIMIT_VIOLATION or SOFT_LIMIT_VIOLATION
- no special color
  for all other cases

8.3.2.3.2.6 Graph Facility

To call the Graph Facility, select

Online_Test_Control –> Data Displays –> Graph Facility

To animate the Graph Facility with data acquired from a test node select the *Properties* button as shown in Figure 8–36. The Graph Facility Properties dialog will then pop up (see Figure 8–37).

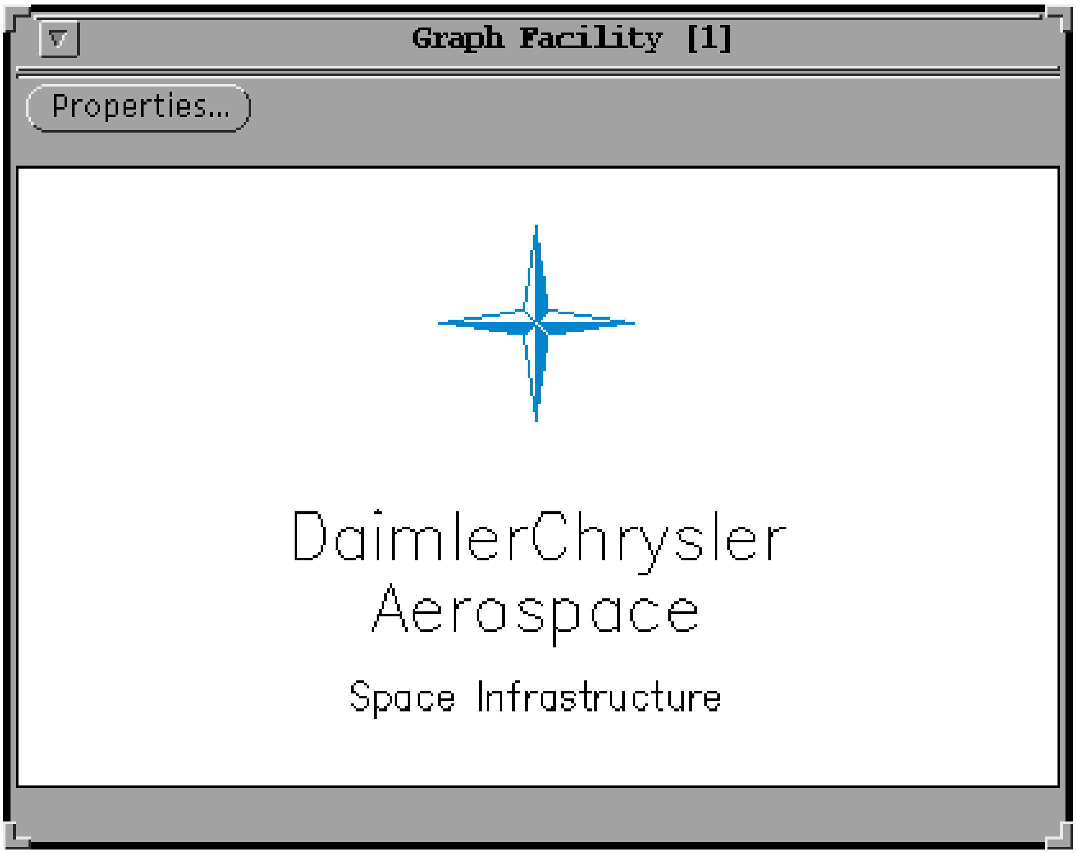


Figure 8–36 : *Graph Facility*

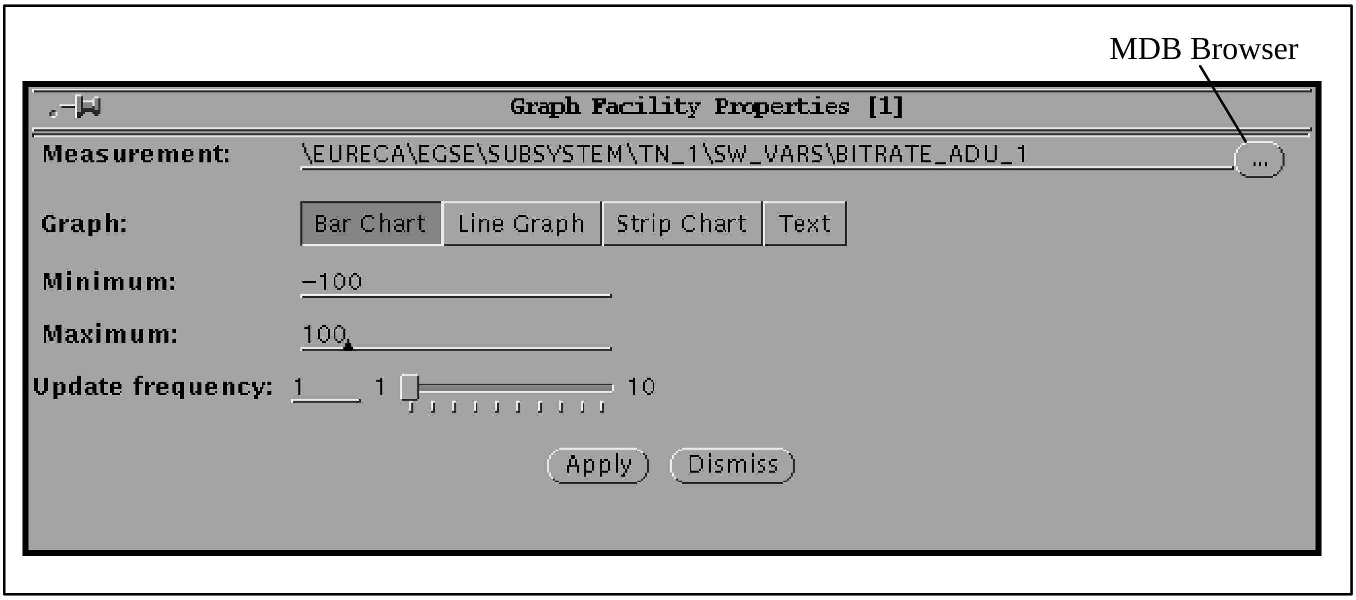


Figure 8–37 : *Graph Facility Properties*

On the Graph Facility Properties dialog you can enter the database pathname of a measurement, derived value, or software variable. In the following the term measurement is used for software variables and derived values as well. If the pathname is very long it could be more convenient to use the Graph Facility's MDB Browser. The MDB Browser is opened by selection of the abbreviated button (labelld "...") behind the measurement input field (Figure 8–38).

With the MDB Browser, you can easily move through the mission database scope. The scope is the Configuration Control Unit (CCU) defined by the actual test configuration. To navigate through the mission database, select a folder (virtual node or CDU) by mouse. To step into (open) a folder, click on the selected folder using the select mouse button. When you found the mesurement to be displayed, select it and press the Apply button. The complete pathname is then inserted in the measurement input field on the Properties dialog.

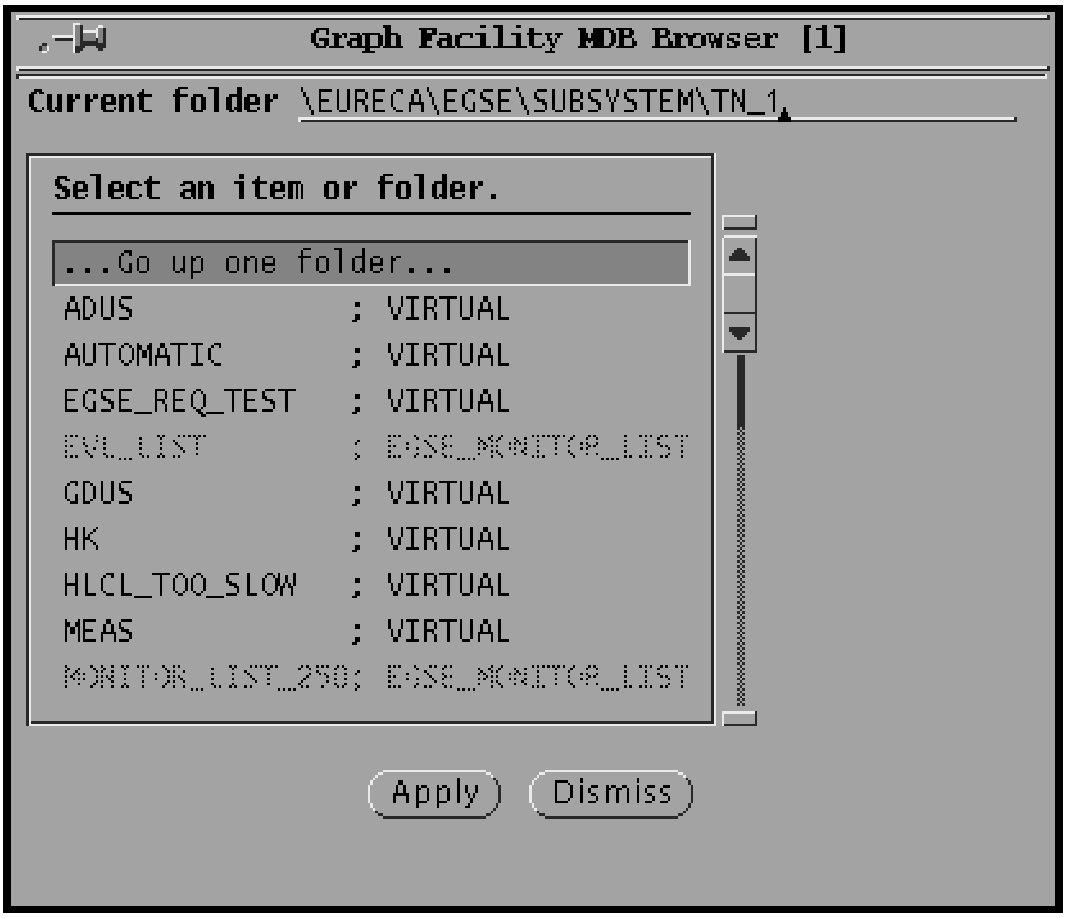


Figure 8–38 : *Graph Facility MDB Browser*

With the Graph choice on the Properties dialog you can select how the measurement is displayed. The Online Test Control provides for types of graphs:

- Bar Chart
  Draws one bar, able to display integer and float measurements.
- Line Graph
  Draws a line starting at the left edge of the graph, able to display integer and float measurements.
- Strip Chart
  Plots a line graph that begins at the right edge of the graph and scrolls toward the left of the graph. The most recent value appears at the right edge of the graph and the history shifts continually to the left. Able to display integer and float measurements.
- Text
  Displays text. Able to display text and discrete measurements as well as float and integer.

The Minimum and Maximum values entered on the properties dialog are used to scale the value axis of the graph, e.g. Figure 8–40 shows a value axis from –5 to 100.

The Update frequency can be specified to determine how often the graph facility gets data from the test node. The values in seconds in range 1 to 10 can either be entered in the input field or selected by moving the slider.

The properties apply after selection of the *Apply* button. The Graph Facility will then check whether the measurement can be used for animation (this may take a few seconds). If the check was successful the measurement is requested from the test node and the graph is animated. The actual time stamp of the most recent data is displayed on the right footer after each update.

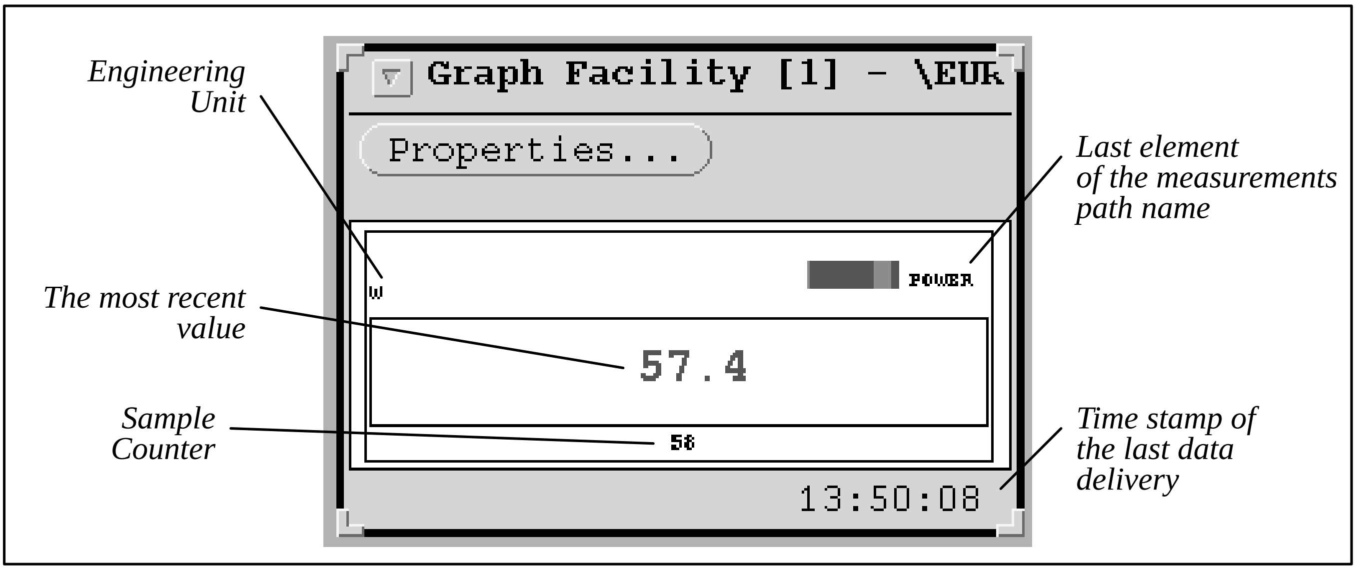


Figure 8–39 : *Bar Chart*

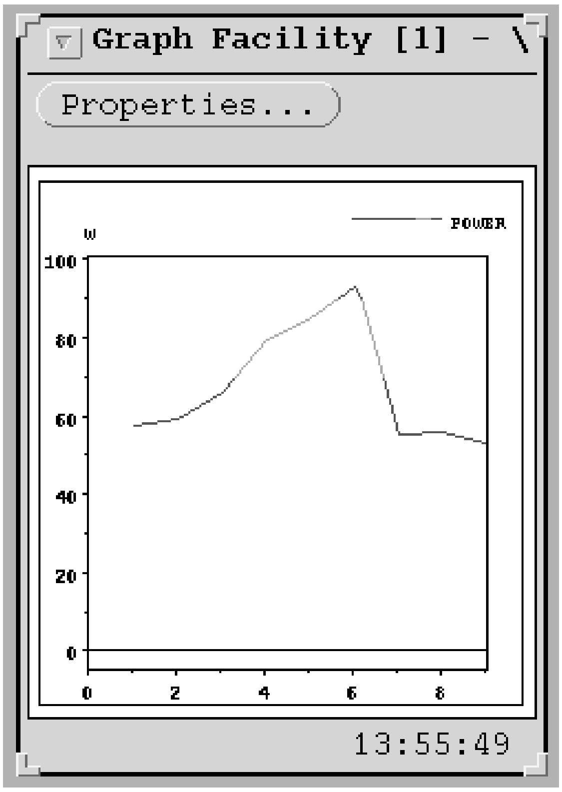


Figure 8–40 : *Strip Chart*

Configuration

The Graph Facility configuration allows to add new graphs (e.g. an indicator) or to modify the existing ones (e.g. change the number of samples in the bar chart from 1 to 25).

REMARK: Before modifying the Graph Facility configuration perform a backup of the directory $GSAF_HOME/hci/config.

The configuration of the Graph Facility can be done in the GRAPH_FACILITY group of hci.ini (see also 8.3.2.5). The next paragraph shows the configuration data as provided with the (CGS) Online Test Control installation (there may be some differences in the layout):

```
[ GRAPH_FACILITY ]
GraphName1=”Bar Chart”
GraphTemplate1=”bar_chart.dv”
GraphName2=”Line Graph”
GraphTemplate2=”line_graph.dv”
GraphName3=”Strip Chart”
GraphTemplate3=”strip_chart.dv”
GraphName4=”Text”
GraphTemplate4=”text.dv”
```

The GraphName*N* attributes define the labels of the graph choice on the properties window (see Figure 8–41), the GraphTemplate*N* attribute determines the template files to be used for animation. For each graph template three template files exist in the $GSAF_HOME/hci/config directory, one for each data type. They are specified by their extensions, `.txt` for text data (byte stream and state code), `.flt` for float data, and `.int` for integer data. E.g., the files for the graph template number 1 (”Bar Chart”) are `bar_chart.dv.txt`, `bar_chart.dv.flt`, `bar_chart.dv.int`.

The following example will describe how to add a new graph, an indicator, to the Graph Facility.

The first step is to append a new graph name and template. Open a text editor (e.g. vi) with the configuration file `$GSAF_HOME/hci/config/hci.ini` and append the indicator definition to the GRAPH_FACILITY group (the bold attributes):

```
[ GRAPH_FACILITY ]
GraphName1=”Bar Chart”
GraphTemplate1=”bar_chart.dv”
GraphName2=”Line Graph”
GraphTemplate2=”line_graph.dv”
GraphName3=”Strip Chart”
GraphTemplate3=”strip_chart.dv”
GraphName4=”Text”
GraphTemplate4=”text.dv”
GraphName5=”Indicator”
GraphTemplate5=”indicator.dv”
```

This will add the new choice for the indicator to the Graph Facility property window as shown in Figure 8–41.

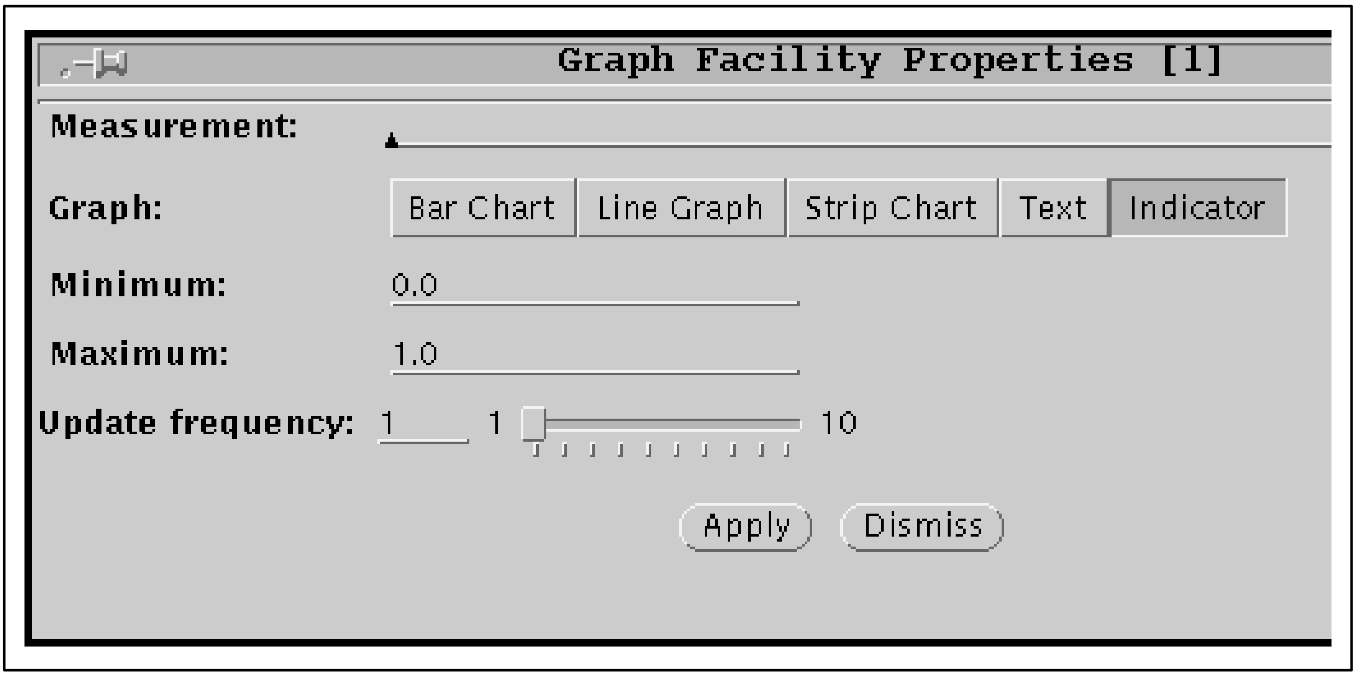


Figure 8–41 : *Property Window with Indicator*

The next step is to add the three templates used for animation for the different data types. Start I_MDB tool and create create three WDU ground synoptic display, e.g. INDICATOR_INT, INDICATOR_FLT, and INDICATOR_TXT (refer to 7.3, Preparing Ground Synoptic Displays). Start GWDU tool and edit INDICATOR_INT: create a graph, select indicator output element, and attach any EGSE integer S/W variable or measurement to it; edit INDICATOR_FLT: create a graph, select indicator output element, and attach any EGSE float S/W variable or measurement to it; edit INDICATOR_TXT: create a text like ”Indicator not available for discrete or byte stream values”.

REMARK:
a) discrete values can only be displayed as text by Graph Facility, i.e. using symbols in Graph Facility templates will lead to unpredicatable results.
b) Don’t forget to set four thresholds, to display danger low, nominal low, nominal high, and danger high violations.

To extract the templates from the database and make them available for Online Test Control, start TSCV tool and activate a test configuration referring the database configuration containing the templates.

Then change to the Online Test Control configuration directory as CGS administrator (to get write permissions)

cd $HCI_HOME/config

and start the utility to extract synoptic display from database (remark: the read_synoptic utility behaves like an Online Test Control, i.e. it can only run on a workstation where a participating Online Test Control can run).

```
$HCI_HOME/util/sun5/read_synoptic
Reading test configuration...
Done.
Opening MDB...
Selecting
```

```
  Element:      EURECA
  Mission:      DUMMY_MISSION
  System tree: \EURECA\EGSE( 3)
  CCU:          TEMPLATES
  Version:       1. 0. 0

Enter pathname (or quit):\eureca\egse\test\md_egse\indicator_flt
Enter filename:          indicator.dv.flt
Writing indicator.dv.flt

Enter pathname (or quit):\eureca\egse\test\md_egse\indicator_int
Enter filename:          indicator.dv.int
Writing indicator.dv.int

Enter pathname (or quit):\eureca\egse\test\md_egse\indicator_txt
Enter filename:          indicator.dv.txt
Writing indicator.dv.txt

Enter pathname (or quit):quit
```

The utilities reads the test configuration, displays the selected database configuration, and prompts for the pathname of the synoptic display and the filename where to write it to.

When starting Online Test Control the next time it is able to display data as indicator as shown in Figure 8–42.

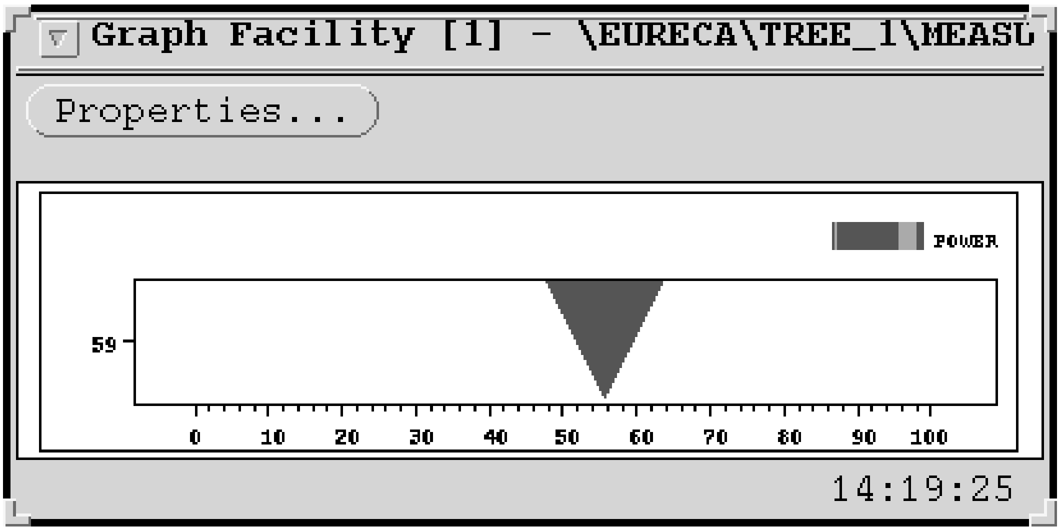


Figure 8–42 : *Indicator*

8.3.2.3.2.7 Monitoring Window

To call the Monitoring Window, select

Online_Test_Control –> Data Displays –> Monitoring Window

In this window user defined items can be monitored. The item information are displayed in a table.

The following information of an end item subdivided into 15 columns is available for the user:

1	Enditem Name	
2	Enditem Description	
3	Time Tag	(the time of the last item value change)
4	Engineering Value	
5	Raw Value	
6	Limit Set	
7	Nominal Limit Low	
8	Nominal Limit High	
9	Danger Limit Low	
10	Danger Limit High	
11	Acquisition Status	
12	Processing Status	
13	Monitoring Status	
14	Delta Monitoring Status	
15	Pathname	(the complete pathname of the end item)

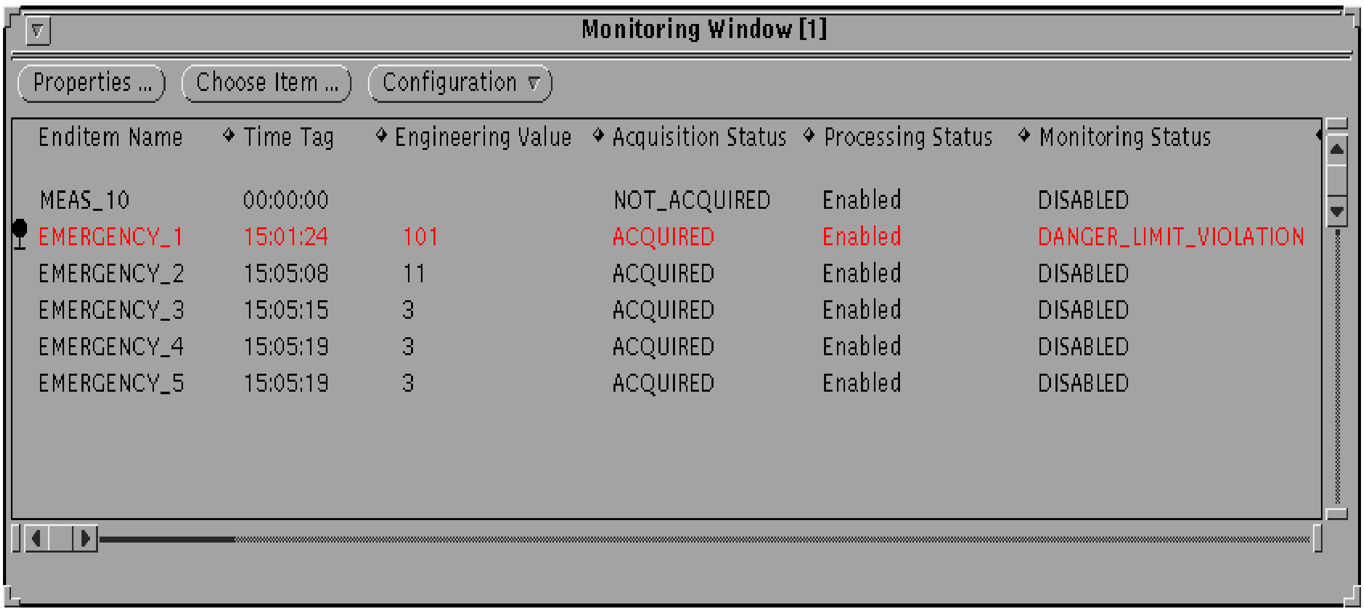


Figure 8–43 : *Monitoring Window*

Adding and deleting items

Press the Choose Item Button on the Monitoring Window to add a single item. The Item Chooser Window pops up where you can navigate through the Database and add end items to the Monitoring Window list. At the top of the window the actual opened folder is shown. Below this text field a list of the database items of the current folder is shown. Make a double click on a ’virtual’ node to open a new folder or select an end item you want to monitor with a single click. Then press the Choose button and now the end item is added to the Monitoring Window list.

If you do not want to choose an item anymore press the Dismiss button and the Item Chooser disappears.

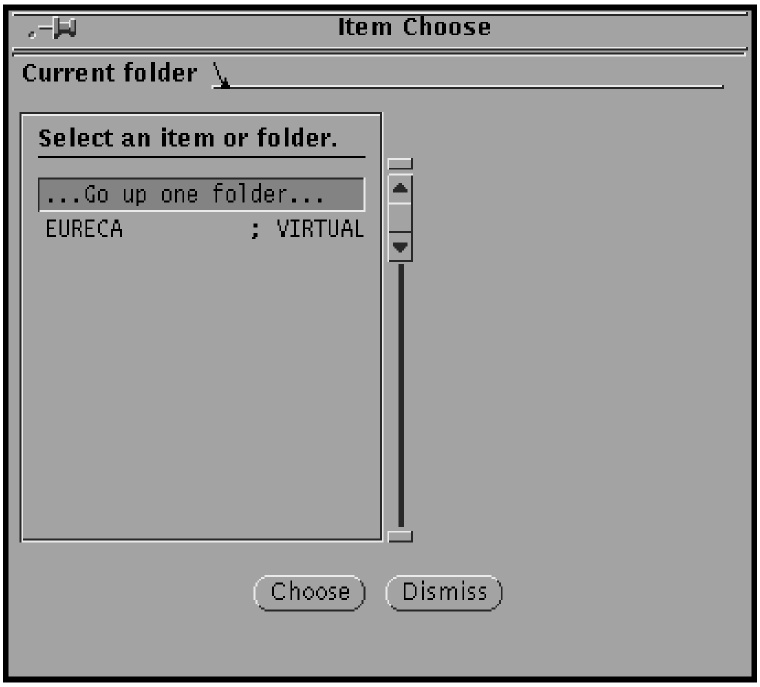


Figure 8–44 : *Monitoring Window Item Chooser*

If you want to delete an item in the Monitoring window list select it with a left mouse button click on the item row. A border around the item row appears. Then press the right mouse button on the selected item row. A menue pops up where you select the delete option. After that the item is deleted from the Monitoring window list.

Save and load of item configurations

If you have a Monitoring Window list of items you want to load later again, select *Confuguration –> Save* and then the Save File Chooser Dialog appears. Go to the folder where you want to save the configuration. Write a filename you want to use to the Save text field and press the Save button. If no error message is reported the configuration is saved. All filenames of Monitoring WIndow Configurations have the postfix “.mon” .

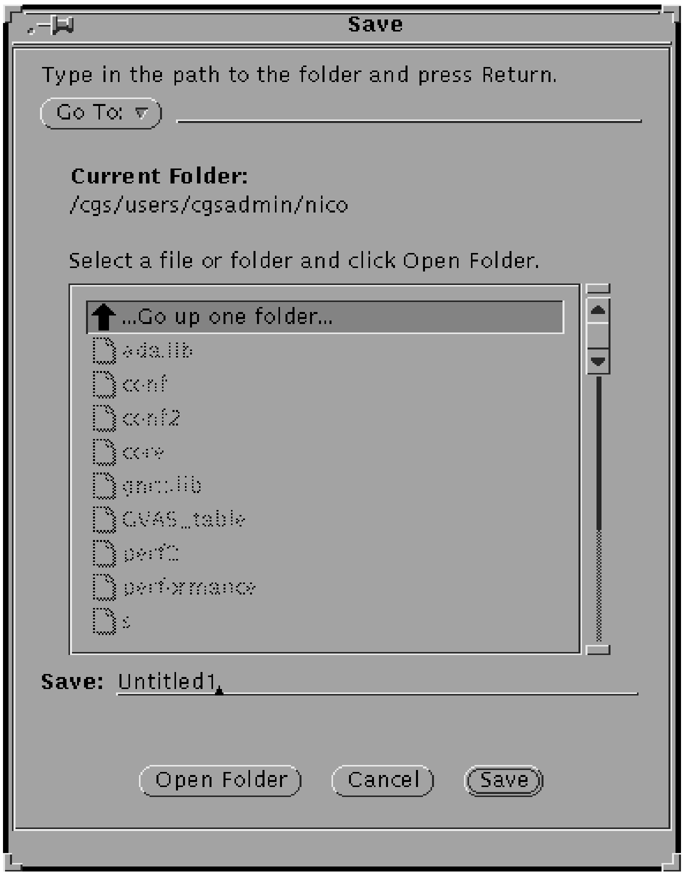


Figure 8–45 : *Monitoring Window Save Configuration*

If you want to load a saved configuration go to the folder where the file is saved, select the file name in the list with a click and then press the Open button. If it is not a valid configuration file a error message is reported else the actual Monitoring Window list is deleted and the new list from the configuration file is loaded.

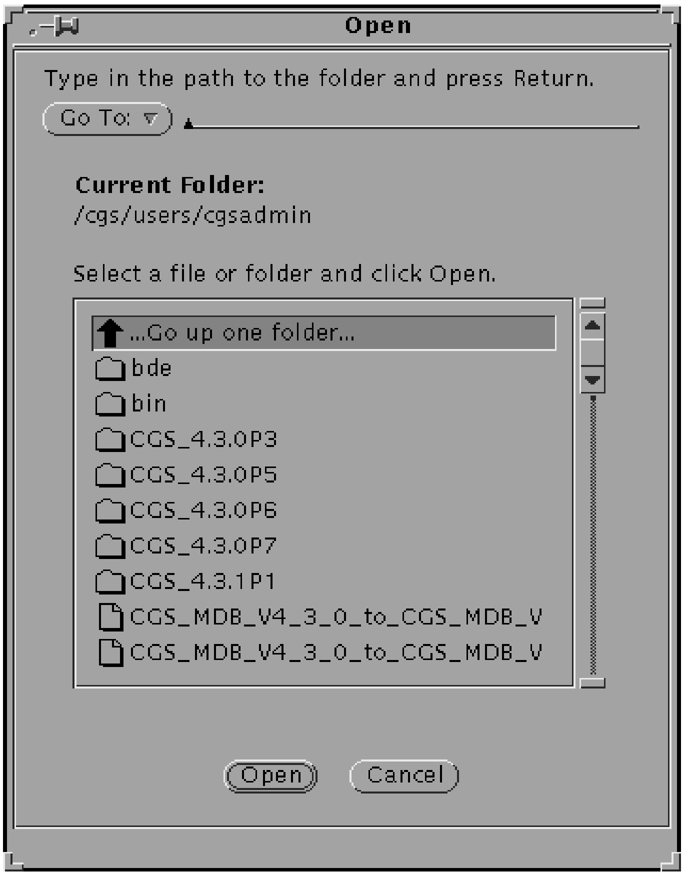


Figure 8–46 : *Monitoring Window Load Configuration*

Properties

The size of all columns can be changed: Select the separator at the end of a column with the left mouse button, keep the button pressed and then drag the separator to a new horizontal position. After releasing the button the column have a new size.

Pressing the Properties button opens the properties window. The column names and their position are shown here. The order of the columns can be changed: Press the button next to the column name with the right mouse button. A menue appears where a new column position can be selected or the column can be disabled, so that it is not displayed anymore in the Monitoring Window.

The raw values can be displayed as binary, decimal or hexadecimal numbers.

Additionally the gaps between columns and rows can be scaled up and down.

Press the Apply button to accept all changes that are made or press the Cancel button to reject them.

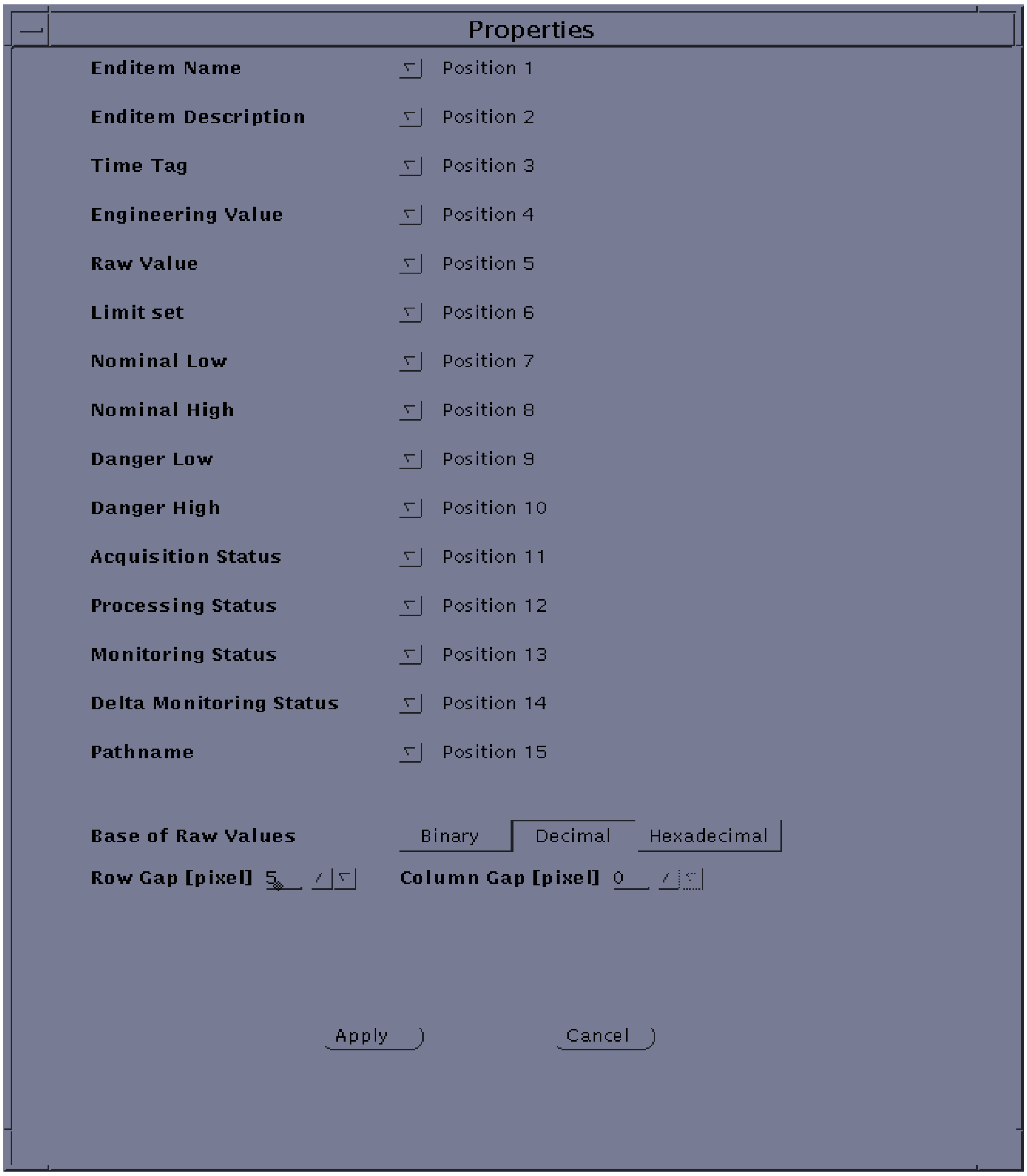


Figure 8–47 : *Monitoring Window Properties*

Colors and Symbols

According to the actual monitoring and delta monitoring status of the end item the row is signed with a special symbol and painted in a special color defined in the hci.ini file:

(Delta) Monitoring Status	Symbol	Color (default)
DISABLED	no symbol	black
IN_LIMITS	√	green
SOFT_LIMIT_VIOLATION		yellow
DANGER_LIMIT_VIOLATION		red

If the window is iconized the background color of the icon is painted in

- red
  if one displayed item is in DANGER_LIMIT_VIOLATION
- yellow
  if one displayed item is in SOFT_LIMIT_VIOLATION and none in DANGER_LIMIT_VIOLATION
- green
  if one displayed item is IN_LIMITS and none in DANGER_LIMIT_VIOLATION or SOFT_LIMIT_VIOLATION
- no special color
  for all other cases

8.3.2.3.2.8 Raw Data Dump

To call a Raw Data Dump window, select
Online_Test_Control –> Data Displays –> Raw Data Dump

The purpose of the Raw Data Dump is to display the content of acquisition data units (ADU).

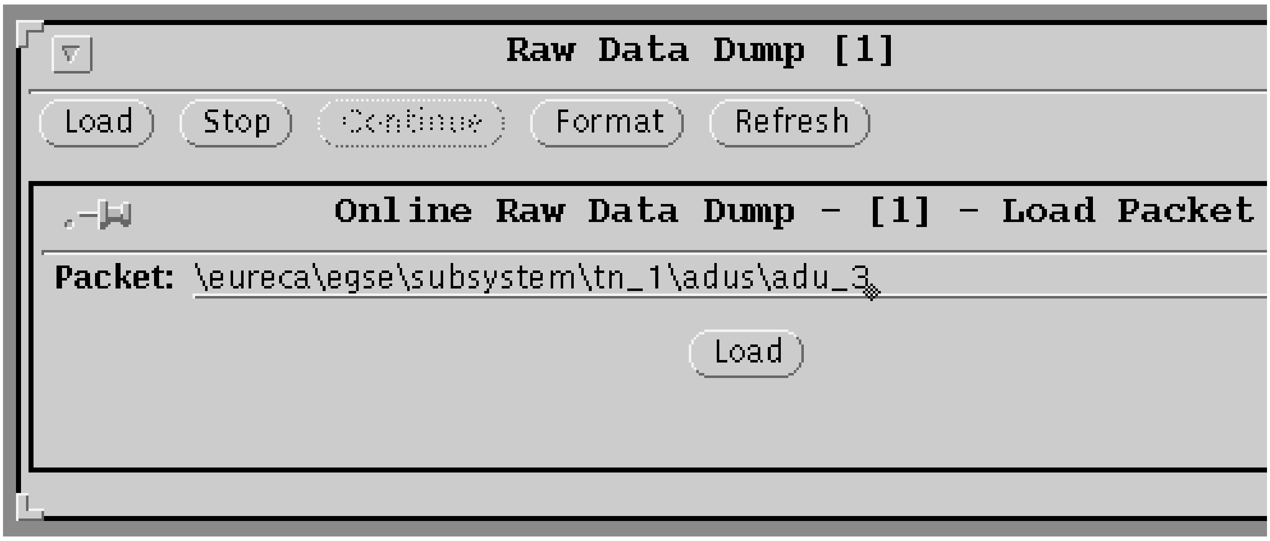


Figure 8–48 : *Raw Data Dump Window (Load Dialog)*

To dump an ADU, select the load button and enter the pathname of the ADU in the Load dialog. If the ADU is available, the raw data is displayed according to its type (unstructured, structured, or CCSDS packet) and format (see Figure 8–50). To change the format (not applicable to structured ADU's), select the format button and choose the new format from the dialog.

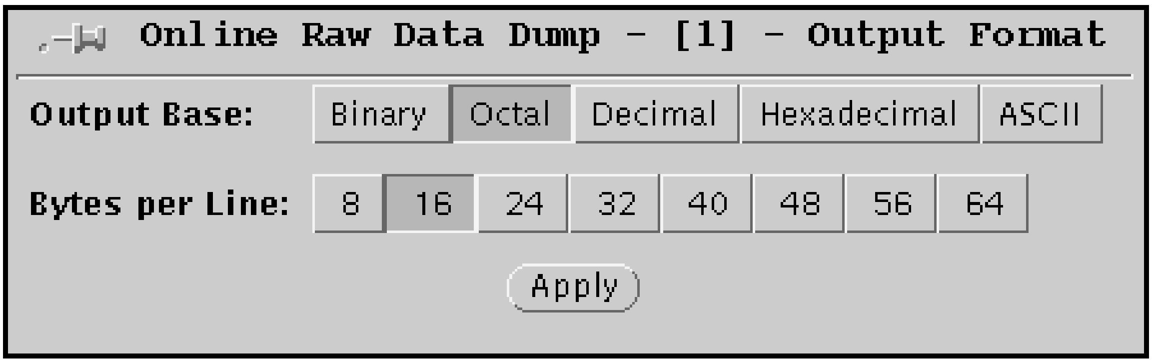


Figure 8–49 : *Raw Data Dump Window (Format Dialog)*

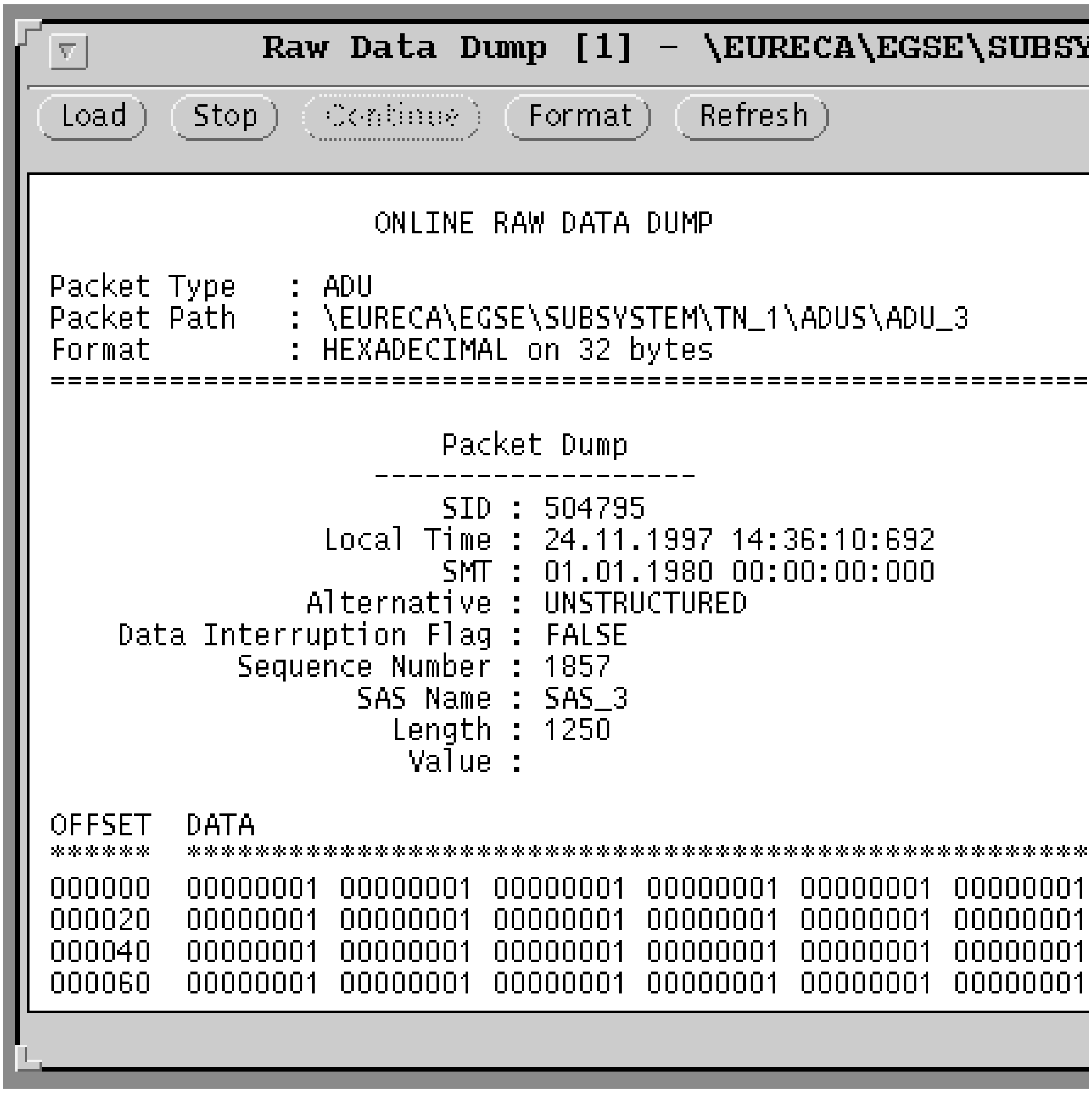


Figure 8–50 : *Raw Data Dump Window*

Remark: When not monitoring the raw data, stop the data display or quit the Raw Data Dump, because updating text windows consumes alot of processing power.

8.3.2.3.2.9 SAS Status

To call a SAS Status window, select
Online_Test_Control –>Data Displays –> *SAS Status*

The SAS Status window shows an overview of the currently executed Specific Application Software (SAS) programs running on a specific test node. A specific test node can be selected like for the AP Status application (see 8.3.2.3.2.1). The name of the test node is shown on the header/title bar of the SAS Status window.

The following items are shown on the window (see Figure 8–51):

- SAS_Name
  The Unix file name of the SAS
- Service
  The service announced by the SAS:

NONE	no service announced
ADU_SERV	the SAS sends ADUs
GDU_SERV	the SAS accepts GDUs
ADU_GDU	the SAS sends ADUs and accepts GDUs

- Messages
  The number of messages sent by the SAS to TES
- Last Message
  Text of last message sent by the SAS. This field is truncated according to the width of the SAS Status window.
- Link–Id
  The Application Identifier of the SAS (i.e. a number assigned by TES)

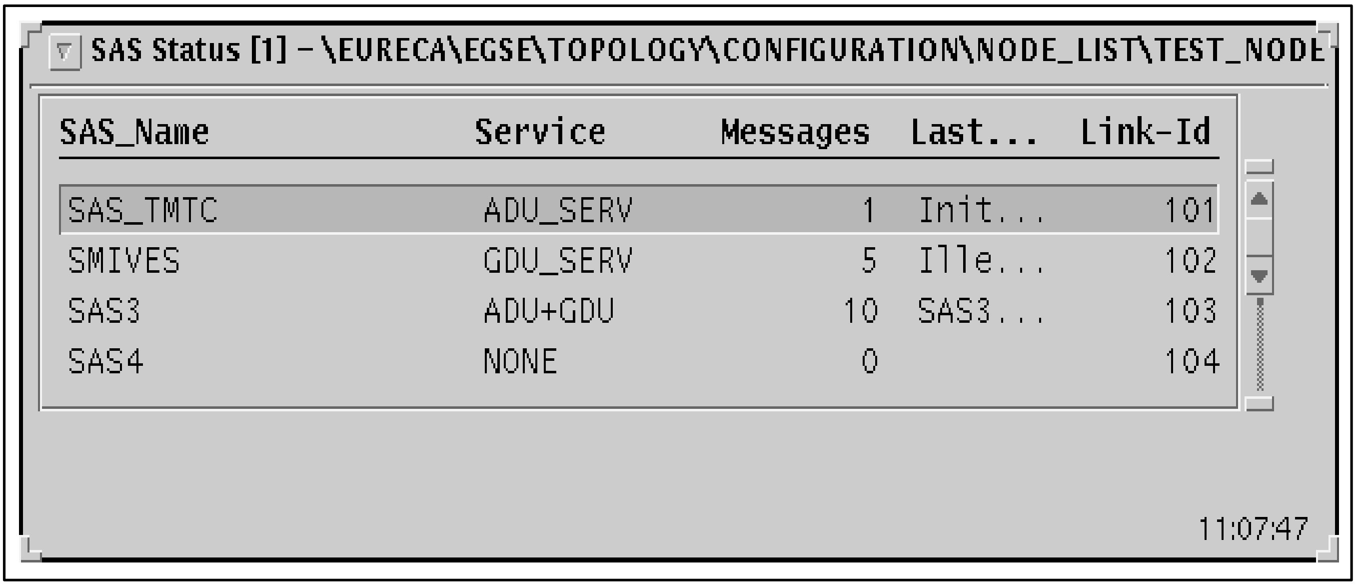


Figure 8–51 : *SAS Status Window*

Abbreviated information can be made visible by clicking on the corresponding item like on the AP Status application (refer to 8.3.2.3.2.1).

8.3.2.3.2.10 System Advisory

To call the System Advisory, select
Online_Test_Control –> Data Displays –> *System Advisory*

The System Advisory window displays the overall status of the test node, the subsystem/unit under test, and the DB server node (see Figure 8–52).

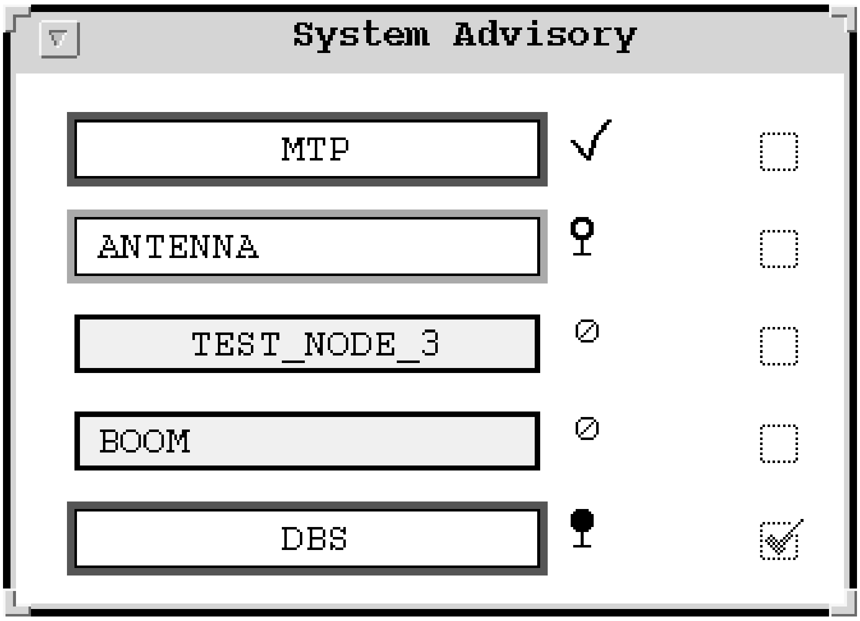


Figure 8–52 : *System Advisory*

The test node boxes are labelled with their name as defined in the mission database and are color–encoded and symbol–encoded with following status indication:

- green and symbol √ :
  The test node status is ok
- yellow and symbol  :
  Not applicable.
- red and symbol  :
  The test node status is error.
- grey and symbol ∅ :
  The test node is not running or doesn’t provide data in time.

By selecting a test node box the corresponding Test Node Status window will be opened.

The subsystem boxes are labelled according to the definition of the EGSE Test Nodes of the EGSE test configuration. To define a label start the I_MDB tool, select an EGSE test node from the test configuration and edit the Subsystem Name attribute (see Figure 8–53).

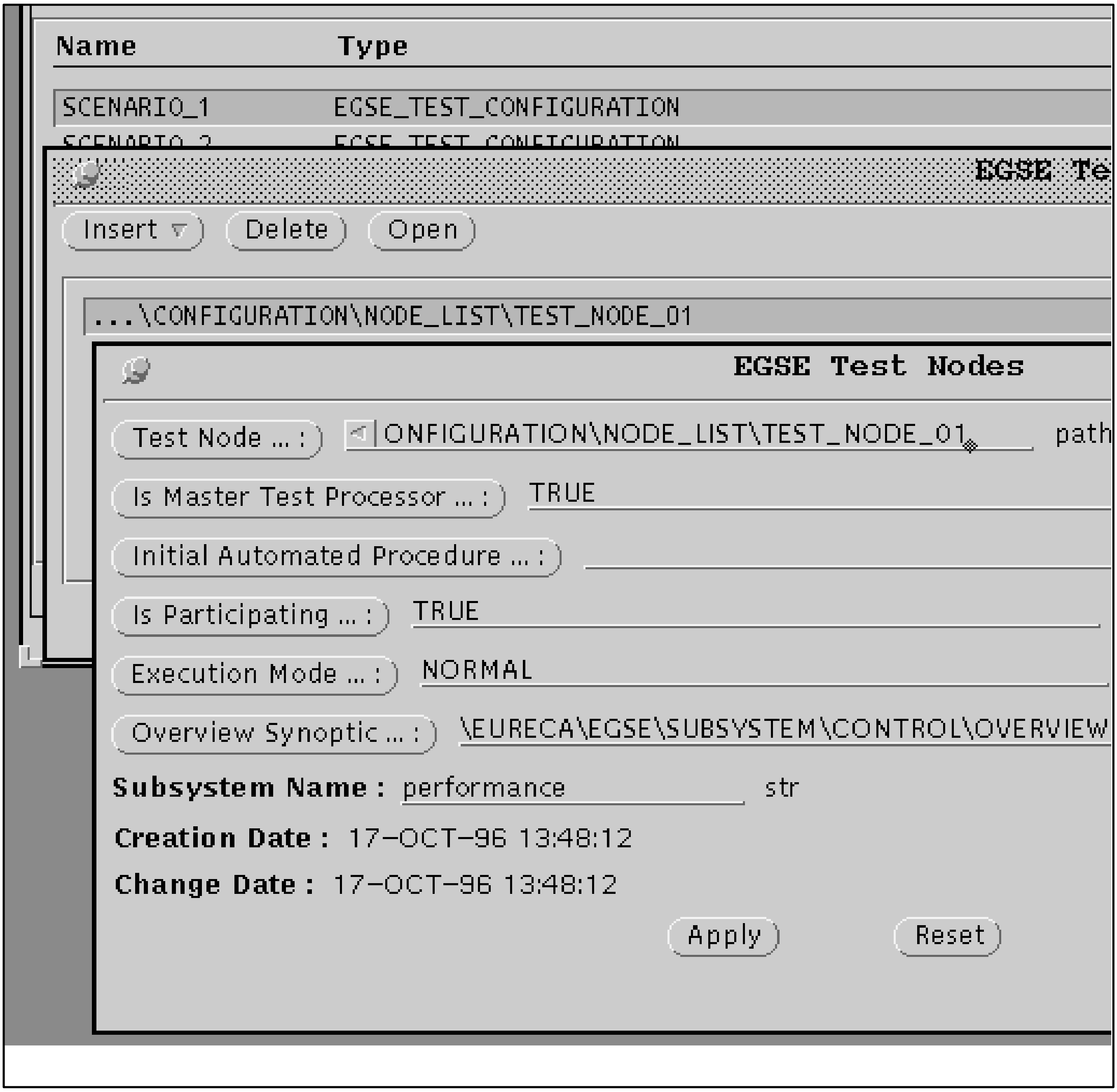


Figure 8–53 : *EGSE Test Nodes Window*

For the subsystem boxes the colors and symbols indicate the following status:

- green and symbol √ :
  All monitored end items are in limits.
- yellow and symbol :
  At least one end item is out of nominal/soft limit.
- red and symbol :
  At least one end item is out of danger/hard limit.
- grey and symbol ∅ :
  No end items are enabled for monitoring.

By selecting a subsystem box the corresponding overview synoptic will be opened. The synoptic can be specified in the same way like the subsystem name (s.a.).

For the database server (DBS) box the following status are applicable:

- green and symbol √ :
  The status of the DBS is ok.
- yellow and symbol :
  The status of the DBS is warning.
- red and symbol :
  The status of the DBS is error.
- grey and symbol ∅ :
  The DBS status can not be acquired from the master test processor.

By selecting the DBS box the Data Base Node Status window will be opened.

If the acknowledge service of the system advisory is enabled a check box is displayed next to each status box. Each time a status box changes to red the corresponding check box is enabled and the user is informed by a beep of the workstations bell to acknowledge the status change. Online Test Control will log each acknowledged alarm.

Configuration

The system advisory can be configured in the SYSTEM_ADVISORY section of hci.ini (see also 8.3.2.5).

To turn off the acknowledge service, set the corresponding attribute to off:

```
[ SYSTEM_ADVISORY ]
AcknowledgeService = Off
```

8.3.2.3.2.11 Explorer

To call the Explorer, select
Online_Test_Control –> Data Displays –> *Explorer*

The Explorer window displays the name tree of the MDB together with status information according to the active test session. It is possible to call UCL commands via menus associated with each name tree element.

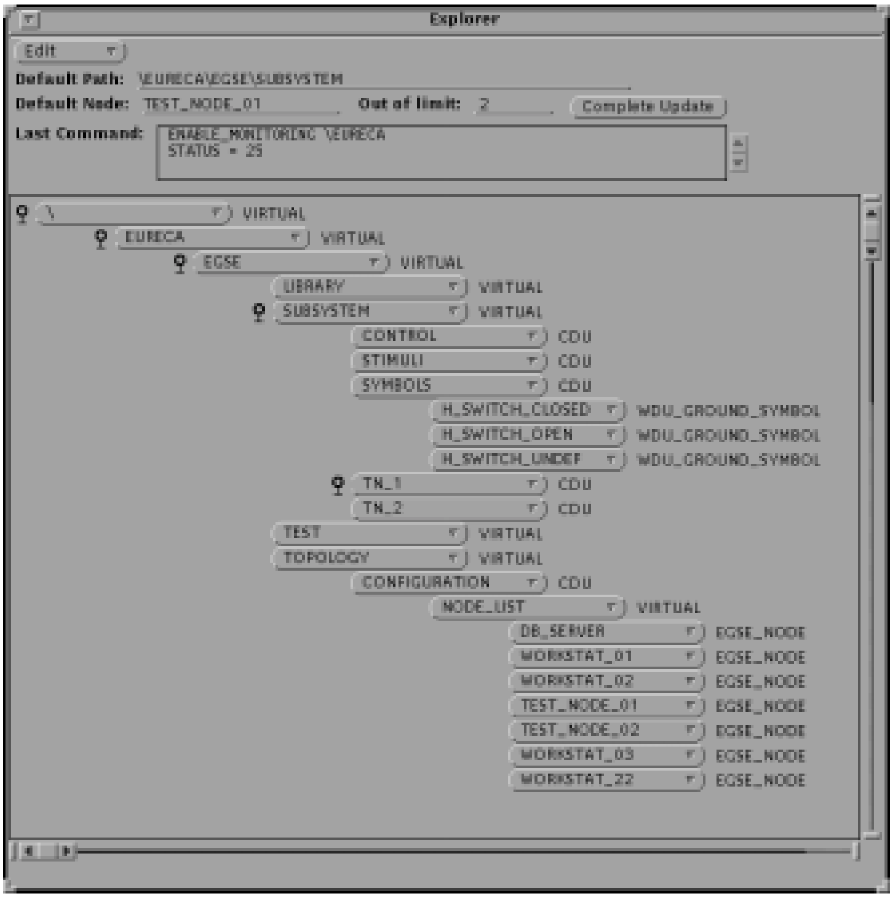


Figure 8–54 : *Explorer Window*

Control Items

At the top of the Explorer window are some control items which make it easier to use the explorer.

- Default Path
  Displays the default path of the Executor’s HLCL interpreter. It could be set by calling the command “Define as default path” on a virtual or CCU node.
- Default Node
  Dispalys the default node of the Executor’s HLCL interpreter. It could be set by calling the command “Set as default node” on a EGSE node.
- Out of limit
  Displays the number of enditems which are out of limit (danger + nominal) on all test nodes.

- Last Command
  Displays the last given command on an enditem and its return parameters.
- Complete Update button
  It should be pressed if an actual and correct status indication of all enditems is needed. But notice, if the button was pressed all data of all enditems are requested at once.
- Edit button
  See section “Properties”.

Navigation and Commands

The nodes of the tree are represented by buttons which are labeled with the node name. On the right side the type of the node is written. On the left side the symbols indicating the status of the node are displayed.

Virtual nodes and CCU’s can be opened or closed by clicking the left mouse button. According to the action the children of a node are displayed or not displayed.

Clicking the right mouse button on a node opens a menu with commands that can be executed on this node. The command and its return parameters are displayed in the “Last Command” text field after executing.

IMPORTANT!

- Most commands are defined in the Ground Library, so it should be imported first. If it is not imported by a login sequence, use the explorer, navigate to the ground library and call the import command there.
- HLCL sequences with interaction should not be executed. The Explorer does not support interaction!
- The Explorer does not support commands with additional in–parameters except for commands that can be called with default parameters.

Status indication

Measurements, variables and derived values (EGSE_xxx_MEASUREMENT, EGSE_xxx_SW_VARIABLE, EGSE_xxx_DERIVED_VALUE) have a status indication:

Priority	Symbol	Meaning
1.		enditem is out of danger limit,
2.		enditem is out of nominal limit,
3.	∨	is in limit
4.	no symbol	enditem is not enabled for monitoring, but acquired
5.	⊖	enditem is not acquired
6.	⊖	the enditem is not available on any of the test nodes participating in the test

Additionally the status indications 1 – 3 are passed on to the parents of the enditem, so it is easy to navigate to an enditem that is out of limit. If there were more children with different status indications the parent gets the status with the highest priority.

Automatic Update

All enditems which are enabled for monitoring at the same time, when the last complete update was running, are automatically updated until the next complete update is started.

A complete update is started if

- the “Complete Update”–Button has been pressed.
- no automatic update is active (after new start) and one ore more enditems are out of limit.

For example, if a disabled enditems has been enabled for monitoring for the first time the actual monitoring status is not displayed automatically. So the “Complete Update”–Button have to be pressed, if a correct display is wanted.

Properties Window

To open the Properties Window select
Edit –> Properties

Available properties settings are

- Command Window Rows
  The number of displayed rows in the text field “Last Command” can be set in a range 1 – 10. The default value is 3.

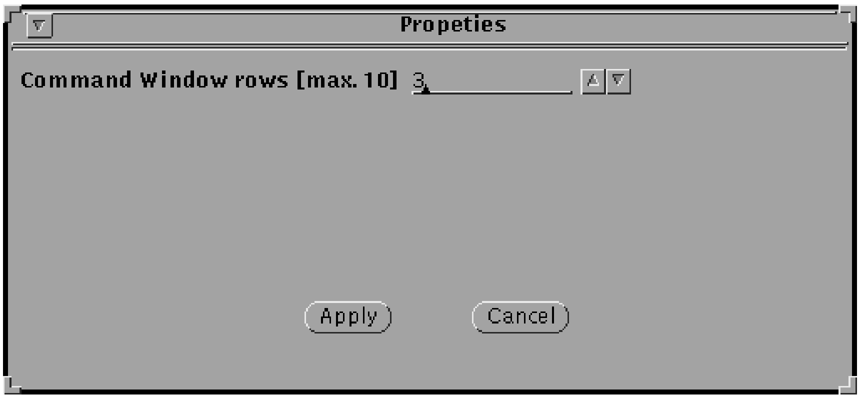


Figure 8–55 : *Explorer Properties Window*

List of Commands

VIRTUAL/CDU
Start_Acquisition, Stop_Acquisition
Enable_Monitoring, Disable_Monitoring
Enable_EVL, Disable_EVL
Withdraw_Condition
Set_As_Default_Path

EGSE_NODE

Enable_Archiving, Disable_Archiving, Close_Archive
Withdraw_All_Conditions
Set_as_default_Node

EGSE_TEST_CONFIGURATION
Start_SMT, Stop_SMT

EGSE_INTEGER_MEASUREMENT
Enable_EVL, Disable_EVL
Start_Acquisition, Stop_Acquisition
Enable_Monitoring, Disable_Monitoring
Get_Integer, Show (=”?”)
Get_Enditem_Monitoring_Status
Get_Full_Enditem_Monitoring_Status
Enable Condiitons, Disable Conditions
Get_Processing_State, Get_Acquisition_Status
Withdraw_Condition

EGSE_FLOAT_MEASUREMENT
Enable_EVL, Disable_EVL
Enable_Monitoring, Disable_Monitoring
Start_Acquisition, Stop_Acquisition
Get_Float, Show (=”?”)
Get_Enditem_Monitoring_Status
Get_Full_Enditem_Monitoring_Status
Enable Condiitons, Disable Conditions
Get_Processing_State, Get_Acquisition_Status
Withdraw_Condition

EGSE_DISCRETE_MEASUREMENT
Enable_EVL, Disable_EVL
Enable_Monitoring, Disable_Monitoring
Start_Acquisition, Stop_Acquisition
Get_Statecode, Show (=”?”)
Get_Enditem_Monitoring_Status
Get_Full_Enditem_Monitoring_Status
Get_Processing_State
Get_Acquisition_Status
Withdraw_Condition

EGSE_BYTE_STREAM_MEASUREMENT
Enable_EVL, Disable_EVL
Enable_Monitoring, Disable_Monitoring
Start_Acquisition, Stop_Acquisition
Get_Byte_Stream , Show (=”?”)
Get_Enditem_Monitoring_Status
Get_Full_Enditem_Monitoring_Status
Get_Processing_State
Get_Acquisition_Status

Withdraw_Condition
EGSE_MONITOR_LIST
Enable_EVL, Disable_EVL
Enable_Monitoring, Disable_Monitoring
Start_Acquisition, Stop_Acquisition
EGSE_INTEGER_SW_VARIABLE
see EGSE_INTEGER_MEASUREMENT
EGSE_FLOAT_SW_VARIABLE
see EGSE_FLOAT_MEASUREMENT
EGSE_DISCRETE_SW_VARIABLE
see EGSE_DISCRETE_MEASUREMENT
EGSE_BYTE_STREAM_SW_VARIABLE
see EGSE_BYTE_STREAM_MEASUREMENT
EGSE_INTEGER_DERIVED_VALUE
see EGSE_INTEGER_MEASUREMENT
EGSE_FLOAT_DERIVED_VALUE
see EGSE_FLOAT_MEASUREMENT
EGSE_DISCRETE_DERIVED_VALUE
see EGSE_DISCRETE_MEASUREMENT
EGSE_STRING_DERIVED_VALUE
see EGSE_BYTE_STREAM_MEASUREMENT
EGSE_ANALOG_STIMULUS
Enable_Enditem, Disable_Enditem,
Issue
EGSE_DISCRETE_STIMULUS
Enable_Enditem, Disable_Enditem,
Issue
EGSE_BINARY_PACKET
Enable_Enditem, Disable_Enditem,
Issue
EGSE_PREDEFINED_TC
Enable_Enditem, Disable_Enditem,
Issue
GDU_DESCRIPTION_LIST
Enable_Enditem, Disable_Enditem,
Issue
CCSDS_ADU_DESCRIPTION
Start_Acquisition, Stop_Acquisition
Set_Bits_In_Simulated_ADU, Send_Simulated_ADU
STRUCTURED_ADU_DESCRIPTION
Start_Acquisition, Stop_Acquisition
Set_Bits_In_Simulated_ADU, Send_Simulated_ADU
UNSTRUCTURED_ADU_DESCRIPTION
Start_Acquisition, Stop_Acquisition
Set_Bits_In_Simulated_ADU, Send_Simulated_ADU
UCL_AUTOMATED_PROCEDURE
Execute
Load_UCL

Get_ID
UCL_SYSTEM_LIBRARY
Import
UCL_USER_LIBRARY
Import
WDU_GROUND_SYNOPTIC_DISPLAY
Display_picture

8.3.2.3.2.12 Test Node Status

To call the Test Node Status window, select
Online_Test_Control –> Data Displays –> *Test Node Status*

The Test Node Status window shows an overview of some housekeeping values of a test node. The test node can be selected by the Test Nodes submenu; the MTP is used as default.

The status information displayed on the Test Node Status is grouped into six categories. The category currently displayed can be selected via the status choice list.

- General (see Figure 8–56)
  Shows general status information
  - Test Configuration
    The value shows the pathname of the test configuration used to setup the EGSE.
    It determines which data have been loaded by the test node, too.
  - MTP
    Indicates whether the test node is configured as Master Test Processor (MTP) or Special Check Out Equipment (SCOE) node.
  - Current Mode
    Current test execution mode

NONE	No mode was selected
NORMAL	The test node is connected to the normal data acquisition and data generation links of the test object
REPLAY	The test node replays data from previously archived data.
SIMULATE	The test node simulates data acquisition and data generation links: Measurement data is generated and Telecommands/Stimuli are internally accepted, but not sent.

  - Status
    Current status of the test node

IDLE	The test node is idle.
RUNNING	The TES software on the test node is active and accepts UCL commands / data requests.
SUSPEND	The TES software running on the test node is suspended. No other commands than a mode switching command is allowed.
ERROR	An error occured in the TES software running on the test node

  - Active APs
    The number of active Automated Procedures executed on the test node.
  - Suspended APs
    The number of suspended Automated Procedures loaded on the test node.

- State
  Indicates whether archiving is enabled.
- Close Cycle
  The cycle time in minutes an archive file will be closed automatically.
- Free Disk Space
  Free disk disk space of the test node’s local disk in kilo bytes.

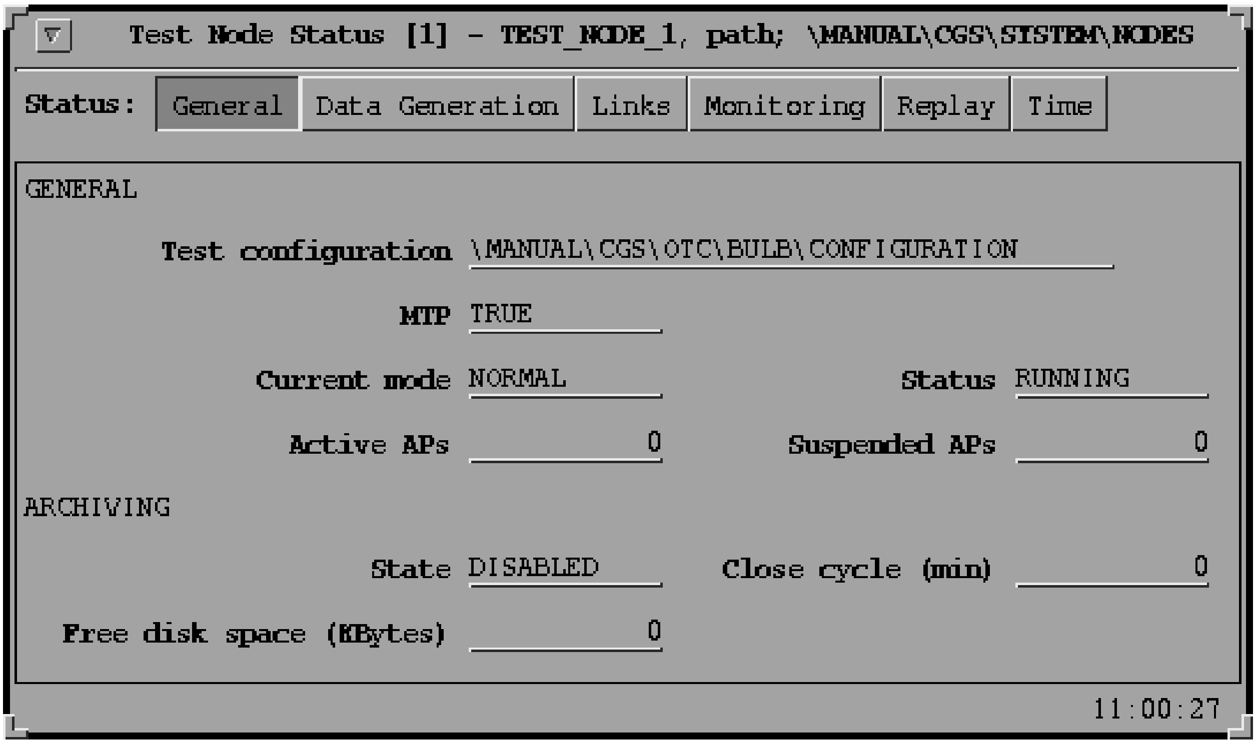


Figure 8–56 : *Test Node Status – General*[1]

1. To get the snapshot of the whole window into this documentation the font size of Online Test Control has been reduced to Helvetica 8.

- Data Generation
  Displays statistics about Generation Data Units (GDUs) loaded from the data base and the current status (see Figure 8–57).

  LOADED FROM DATABASE

  - Total GDUs
    The total number of GDUs loaded from database.
  - CCSDS TC GDUs
    Number of GDUs with CCSDS (Consultative Committee for Space Data Systems) Tele–Commands loaded from database.
  - Digital GDUs
    Number of digital output GDUs loaded from database.
  - Analog GDUs
    Number of analog output GDUs loaded from database.
  - Bin. pack. GDUs
    Number of binary packet GDUs loaded from database.

  CURRENT STATUS

  - Stimuli sent out
    Number of stimuli sent out since last start (includes erroneous one's).
  - Stimuli errors
    Number of GDUs with errors in Special Application Software (SAS) since last start.
  - Last error in
    Pathname of the last stimulus that resulted in an error.
  - at SAS
    Name of the Special Application Software reporting the last error.

  VERIFICATION

  - Successful
    Number of GDUs sent with successful verification.
  - Failed
    Number of GDUs sent with failed verification.

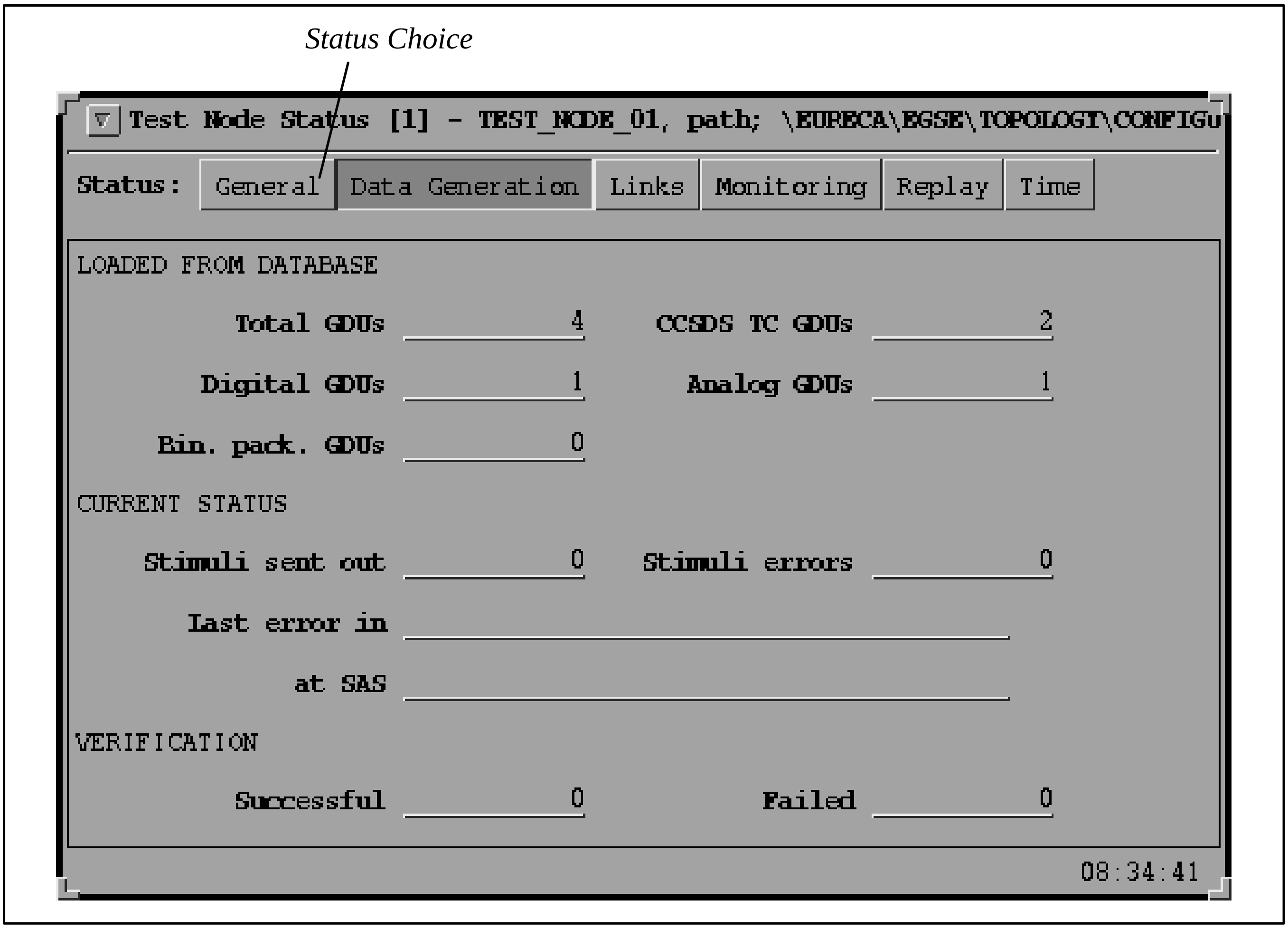


Figure 8–57 : *Test Node Status – Data Generation*

- Links (see Figure 8–58)
  - Connected workstations
    The number of workstations (Online Test Control software) connected to the test node.

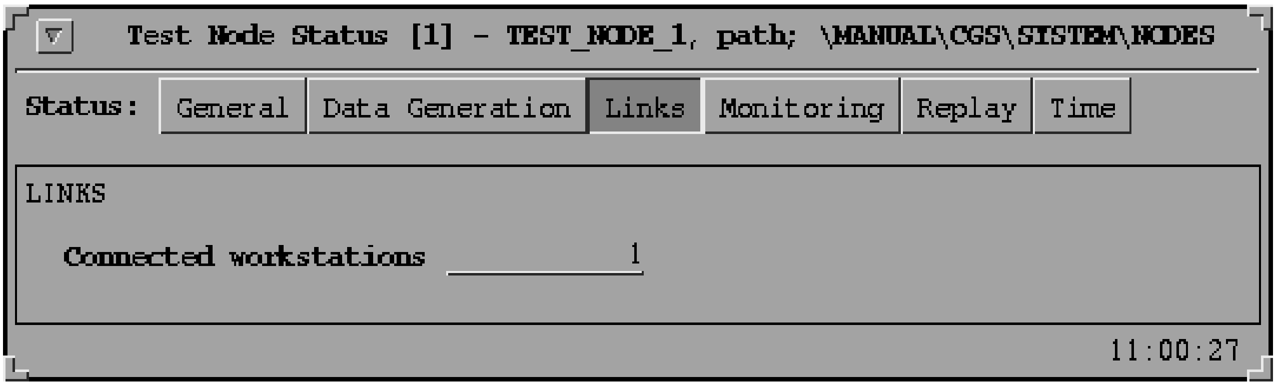


Figure 8–58 : *Test Node Status – Links*

- Monitoring (see Figure 8–59)

  LOADED FROM DATABASE

  - Monitorable Enditems
    Total number of enditems that can be monitored.
  - Discrete
    Number of discrete enditems that can be monitored.
  - Analog
    Number of analog enditems that can be monitored.
  - Bytestream
    Number of bytestream enditems that can be monitored.
  - Measurements
    Number of measurements loaded from MDB.
  - SW variables
    Number of measurements loaded from MDB.
  - Derived values
    Number of derived value definitions loaded from MDB.

  CURRENT STATUS

  - Enabled
    The number of enditems enabled for monitoring.
  - Acquired
    Number of measurements currently acquired.
  - with EVL
    Number of acquired measurements or software variables with EVL.
  - Out of soft limits
    Number of enditems currently out of soft/nominal limit.

- Out of soft limits/since last start
  Number of enditems out of soft/nominal limit since last start.
- Out of danger limits
  Number of enditems currently out of danger/hard limit.
- Out of danger limits/since last start
  Number of enditems out of danger/hard limit since last start.

CONDITIONS

- Defined
  Number of conditions defined.
- Enditems
  Number of enditems carrying conditions.
- Actions triggered
  Number of actions triggered from conditions since last start.

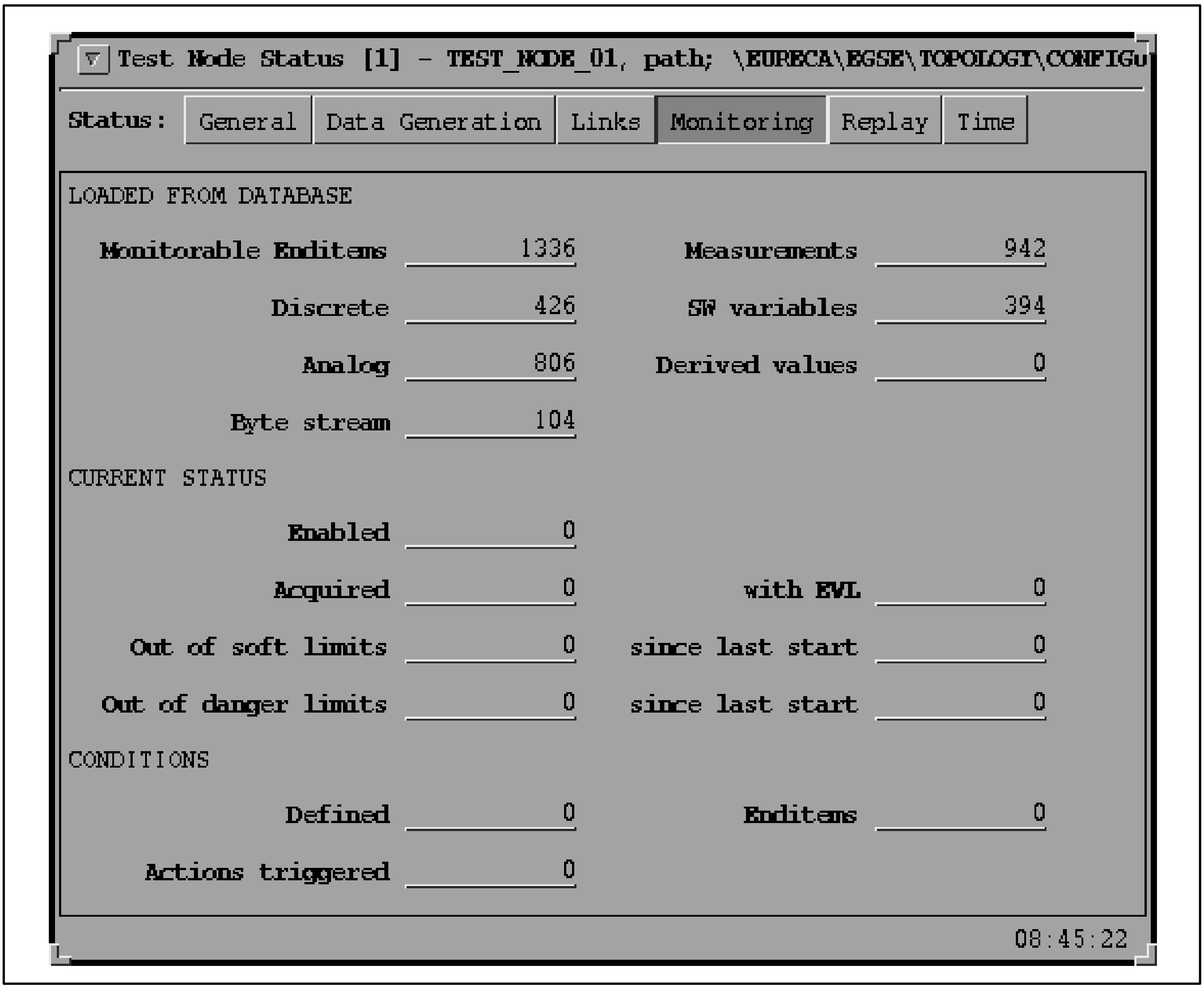


Figure 8–59 : *Test Node Status – Monitoring*

- Replay Simulation
  - Speed
    Replay speed in percent.
  - Begin time
    Replay begin time selected during initialization.
  - End time
    Replay end time selected during initialization.

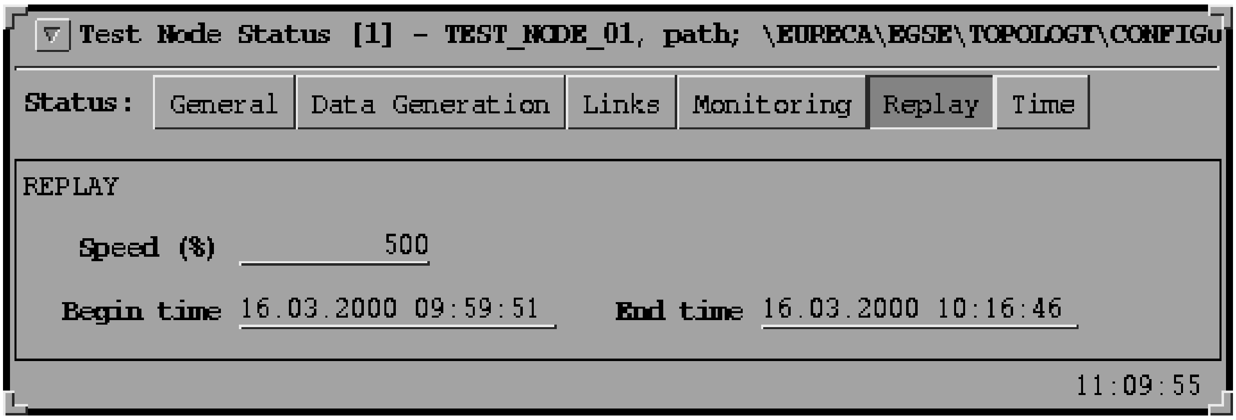


Figure 8–60 : *Test Node Status – Replay*

- Time
  - Clock synchronized
    Indicates whether the local clock of the test node is synchronized with the NTP server.
  - SMT status
    Indicates the status of the Simulated Mission Time.
  - External MTU
    Indicates whether the test node has an external master time unit.

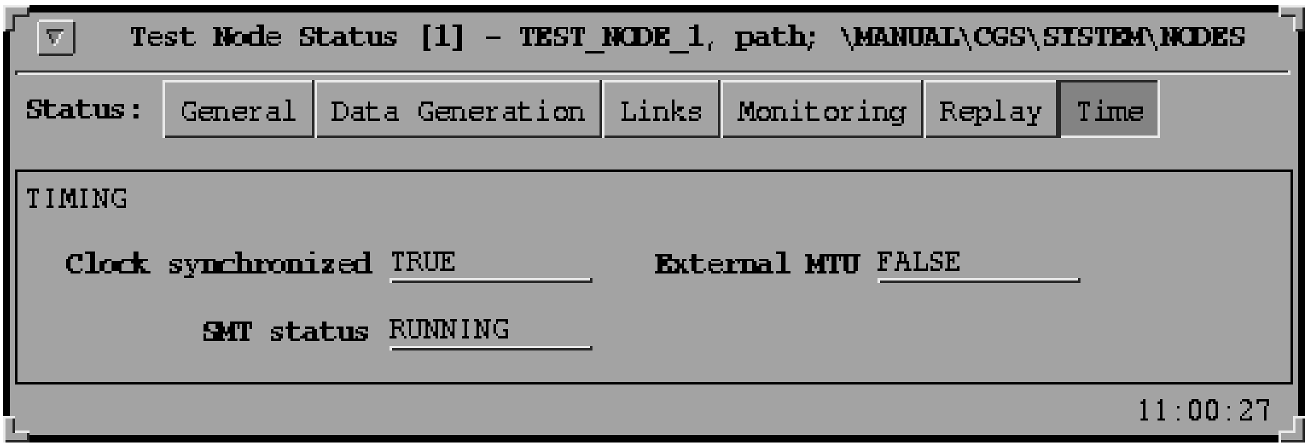


Figure 8–61 : *Test Node Status – Time*

Configuration

The size of the test node status is automatically adapted to the selected status group. To disable auto resize set AutoResize to off in file hci.ini (s.a.).

```
[ TEST_NODE_STATUS ]
AutoResize = Off
```

8.3.2.3.2.13 Synoptic Displays

Synoptic displays can't be called via Online Test Control Menu, but via the HLCL command ASSIGN_PICTURE, the UCL library procedure DISPLAY_PICTURE, or if loaded via a screen setup.

A synoptic display animates pictures with measurement data (calibrated values) received in VICOS. Pictures are defined during Test Preparation via the GWDU Editor. They are stored in the MDB and loaded by Online Test Control on request.

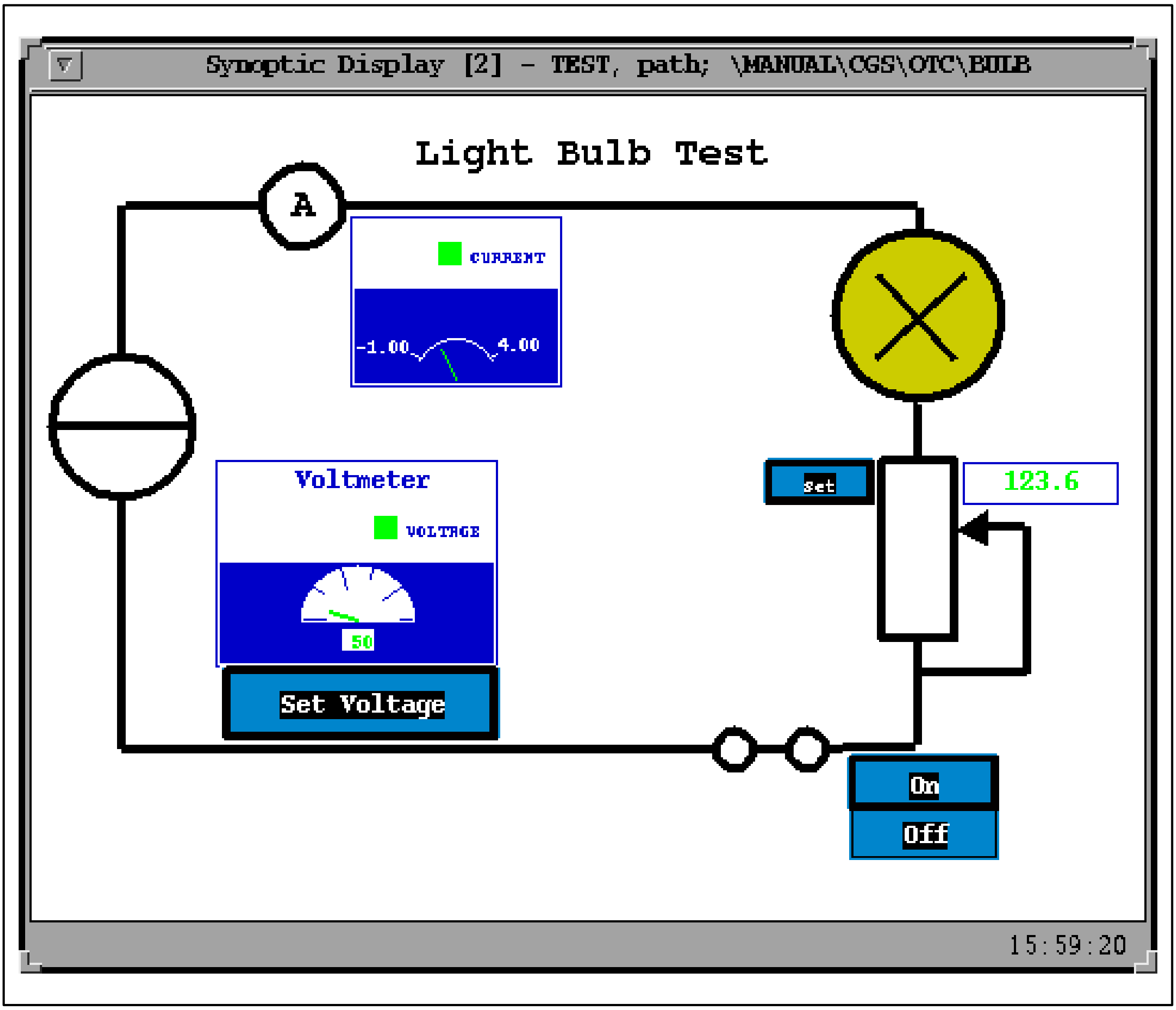


Figure 8–62 : *Synoptic Display*

Synoptic Displays request data from test nodes. Before valid data can be displayed, the test nodes acquisition functions must have been setup correctly. Refer to Test Execution Control Functions

8.3.2.3.2.13.1 Structure of Synoptic Displays

The **title bar** of a synoptic display shows which picture is loaded (enditem name of the picture).

The middle part (**display area**) of the synoptic contains the picture. The pictures can contain static elements such as lines, circles, rectangles, etc., dynamic elements for data display and input elements for user commanding.

Synoptic pictures can be composed of input and/or output elements (to get an overview of the elements, refer to GWDU User's Manual and Operations Manual [2.1.2.1], section 2.2 Object Creation).

Synoptic pictures are 'animated graphics'. That means, the actual appearance of a synoptic display depends on the incoming data. The main use of synoptic pictures is to monitor data values in real-time.

The dynamic parts belong to one of these classes:

- Animation of size
  All kinds of "thermometers" and "bar–graphs " belong to this class.
- Animation of orientation
  All kinds of "pies", clocks and dials belong to this class
- Animation of text and symbols
  All kinds of dynamic textual displays and data–dependent symbol presentation belong to this class.

Furthermore, the first two animation classes may additionally display a history of values, i.e. the change of value over time.

Animation of color provides redundant information: If applicable, the display of nominal values, of values too low and of values too high are distinguishable without perceiving the exact values of the data. Values violating a danger (hard) limit should be displayed using red colors. Values violating a nominal (soft) limit should be displayed using yellow colors. Nominal values should be displayed green.

Static graphical elements are used to refine the layout of a synoptic picture and to put label information to the dynamic output symbols.

The basic **input elements** are the picture markers, buttons and the menus. Input elements allow to generate HLCL commands or to load another picture (see below).

The status is displayed from left to right in the **footer line**. The following status information is possible:

- Ready
  Synoptic definition is loaded and displayed (This does not mean, that data is already delivered).
- Data Requested
  Data is requested for all dynamic elements contained in the synoptic display from the test execution system. If this status information changes to the time display, the requested values for the enditems will be contiguously delivered and the corresponding dynamic element is updated in the display.

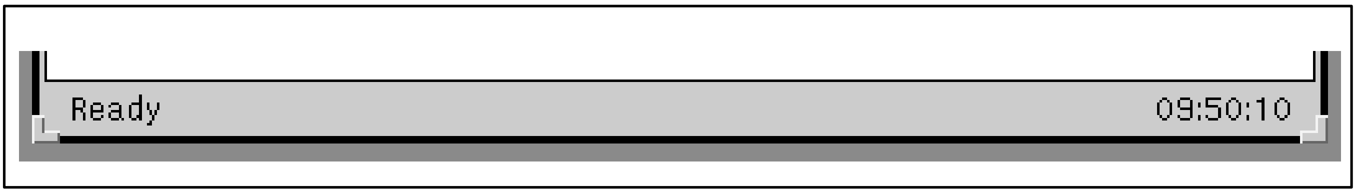


Figure 8–63 : *Synoptic Display Status/Footer Line*

– The time stamp is read from the data delivery message. It is the time given by the test execution system on computation. If the synoptic display contains data acquired from more than one test node, a time range showing the oldest and newest data delivery time stamp is shown. If the synoptic display contains only static or input objects, the message ”No data” is displayed instead of the time stamp.
– Cleaning up
The synoptic display is terminated by selecting the ”quit” entry from the windows menu. Termination means to stop the data delivery from the test execution system to this synoptic at first. The status ”cleaning up” indicates the process of requesting a stop delivery is still in progress. If the data delivery has been successfully stopped the synoptic display is taken away.
– Cant remove synoptic during HLCL execution
The user has selected an HLCL command from the synoptic display. This command has not been finished. Return values might be given. Therefore the user is not allowed to remove the synoptic during command execution.
– Cant execute another command during HLCL execution
This message is displayed when selecting a command while the synoptic display is just executing a command. Wait until the execution of the command has finished before selecting a new command.
– ### Window too small ? Could not draw some parts !
This message is displayed when the window is too small to draw all objects. Resize the window until the message disappears.

8.3.2.3.2.13.2 Picture Change within a Synoptic Display Window

A picture change can be defined by using the picture marker element or a menu: Selection of the marker (a rectangular box) or a menu entry will cause Online Test Control to replace the current synoptic display (within the drawing area) with the requested one (i.e. the one defined in the marker/menu). This means, in opposite to initiating the display of a new synoptic, only the *contents* of the current one will be replaced by a new one. The previously loaded picture will stop execution.

To initiate picture replacement, select an picture marker using the ”select” button of the mouse.

Using picture markers / menus enables the user to implement a picture hierarchy and to navigate through the pictures by selection with the mouse.

8.3.2.3.2.13.3 Commanding from a Synoptic Display Window

Synoptic Picture may contain as active input elements menus or buttons. By selecting a button, the pre–defined HLCL command is executed. Menus allow to execute one of several alternatives.

The predefined commands are executed by a specific HLCL Interpreter running in each synoptic display. Executed commands are logged to the Test Result Database.

To call execution of a command inside a Synoptic Picture
– Click over an input element using the ”select” mouse button. This executes a command if the input element is not a menu.
– If the input element is a menu, it pops up showing one or more commands. Select a command using the ”right” mouse button to execute the command or click outside to dismiss the menu.

In case of incomplete HLCL commands defined in the picture definition, Online Test Control will ask for the missing parameter in a pop–up window. Entering the parameter and selecting the ”OK” button inside the pop–up window will complete the command and execute it. Click on the ”Cancel” button to cancel the command.

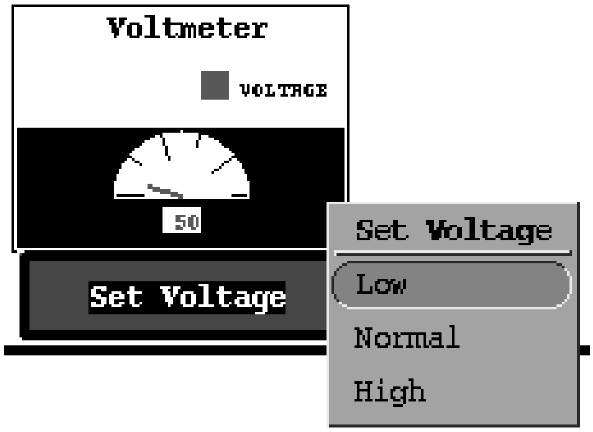


Figure 8–64 : *Figure Figure 8–65 Synoptic Display Popup Menu*

To input EGSE S/W Variables, the HLCL command GET can be used. E.g., to get a resistor value from the user the HLCL command

GET ”Set resistor:”, \MANUAL\CGS\OTC\BULB\RESISTOR

is attached to the “Set” button in the light bulb test synoptic. When the “Set” button was selected, Synoptic Displays will pop up its HLCL Input dialog and the user can enter a appropriate value.

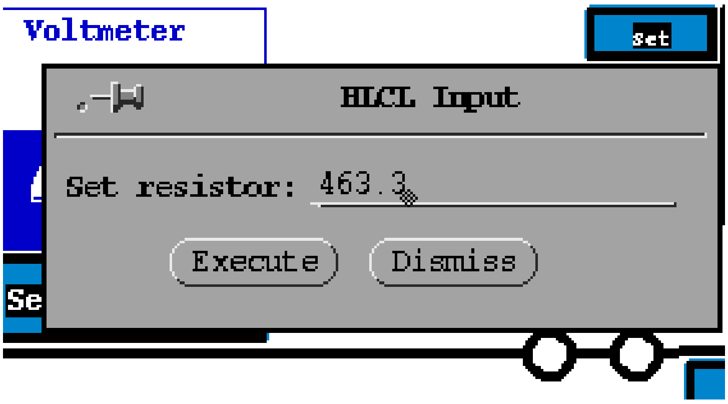


Figure 8–66 : *Figure Figure 8–67 Synoptic Display HLCL Input*

8.3.2.3.2.13.4 Output Objects

<u>Primitive Elements With Dynamic Features</u>

The GWDU Editor lets you add dynamic components to primitive elements like lines, arcs, rectangles, ovals, circles, and polygons that change when the data changes. Dynamic components, called dynamic features, are:

- Attribute dynamics
  Changes attributes such as color, line type, line width, curve type, and arc direction.
- Motion dynamics
  Changes position, rotation, and scale.

- Visibility dynamics
  Changes whether or not the object is visible depending on the data value.
- Text dynamics
  Changes the content of a text object to display the data value.

Figure 8–68 shows a synoptic display containing a circle with motion and color dynamics.

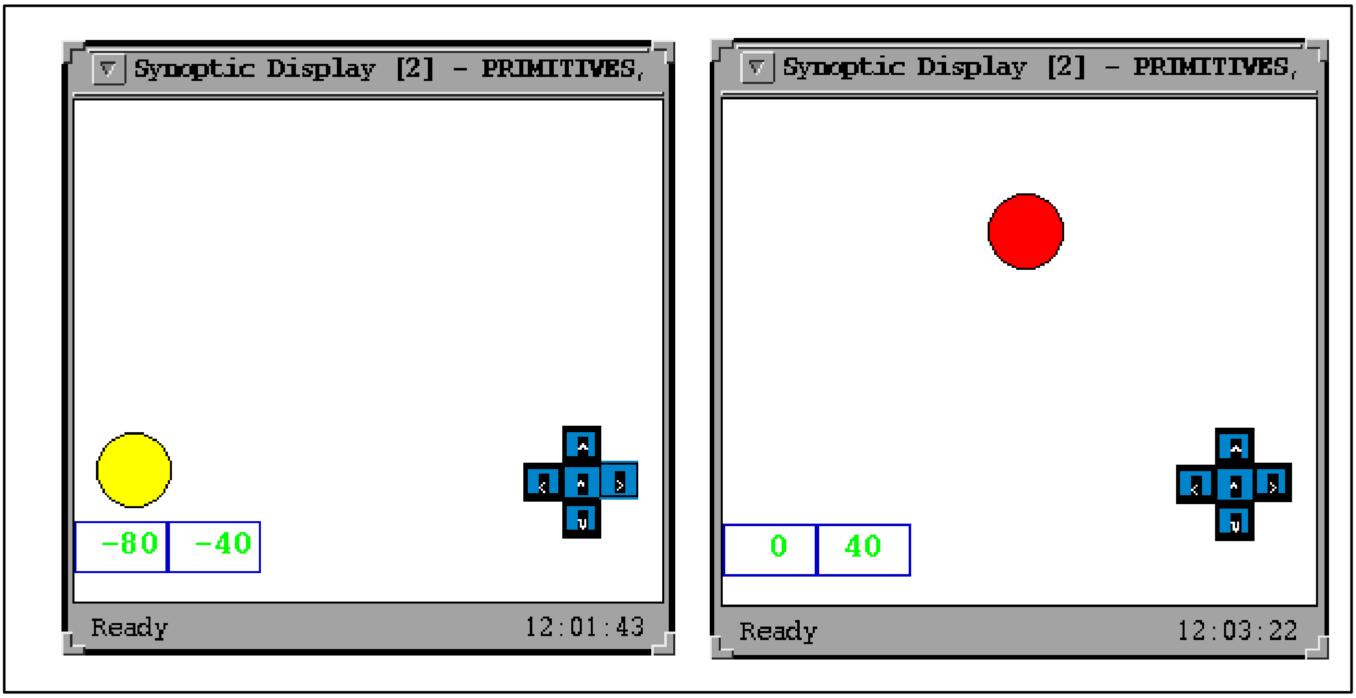


Figure 8–68 : *Primitive Elements With Dynamic Features*

Subdrawings

Subdrawings are driven by discrete measurements or software variables. For every state of the discrete measurement/variable a user–defined symbol is displayed. Figure 8–62 contains such a subdrawing, a switch.

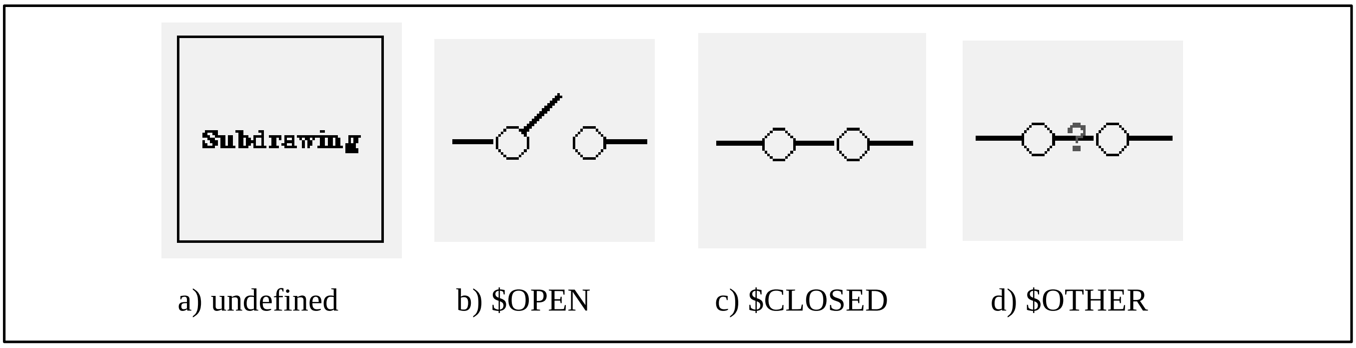


Figure 8–69 : *Subdrawings*

The different symbols acc. to the discrete values are shown in Figure 8–69, Subdrawings.

- a) undefined
  This symbol can not be changed by the user. It is displayed as initial symbol before data are

delivered and whenever data is delivered that does not correspond to a discrete calibration value.

- b) $OPEN and c) $CLOSED
  Symbols displayed according to valid discrete values $OPEN and $CLOSED.
- d) $OTHER
  Default state code, used by the test node in case the raw value can't be calibrated to a state code.

Graph Output Objects

Graph Output objects are pre–defined output objects that display formatted data from measurements and software variables, for example Bar Charts, Line Graphs and Text Graphs. There are many variations of Graph type.

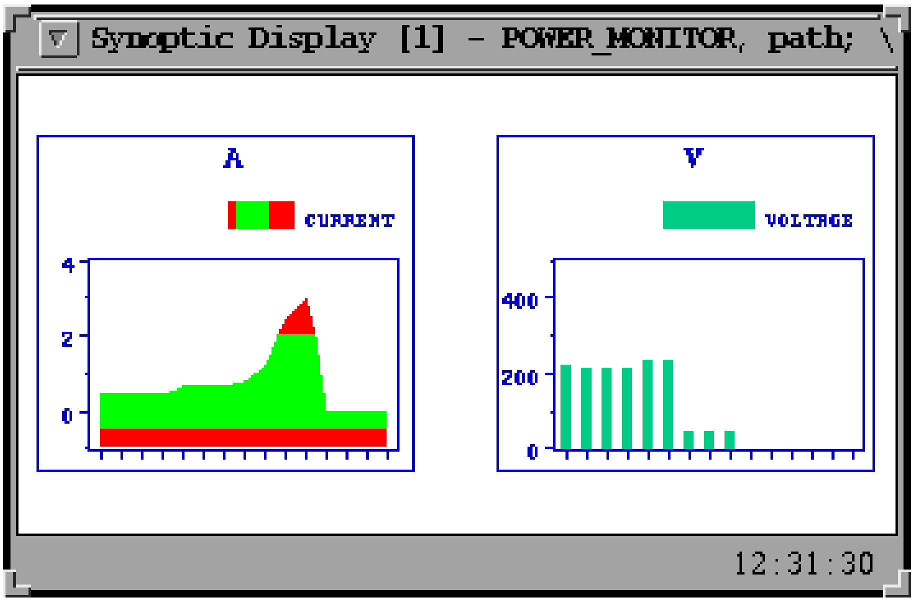


Figure 8–70 : *Graph Output Objects*

Graph Output objects are treated in two different ways, depending whether the thresholds were edited. If the tresholds have been modified using GWDU Editor, no specific processing is done and the thresholds are static. When the thresholds have not been modified the GWDU Editor takes them from the MDB as specified in the danger and nominal limit fields of the measurement/software variable. For those objects, Online Test Control performs an automatic limit handling. When a limit set changes, the corresponding Graph Output objects are reset and the new limits are made applicable. According to the monitoring status and kind of output object, the values are displayed in different color (see Table 8–1). Figure 8–70 shows two such objects; measurement current has danger limits defined and is now in limits (with previous values out of danger limit), for measurement voltage monitoring is disabled.

Monitoring Colors

Monitoring Status	Color of Discrete Values	Color of Analog Values
Disabled	Disabled Color/GWDU Color	Disabled Color
In Limits	GWDU Color	GWDU Color
Nominal (Soft) Limit Violation	Soft Color	GWDU Color
Danger Limit Violation	N/A	GWDU Color

Table 8–1 : *Monitoring Color Mapping*

- Disabled Color : color as defined as configuration parameter MoniroringDisabled (default is “turquoise”).
  If configuration parameter RestoreOriginalColorWhenNotMonitored is set, GWDU Color is used for discrete values.
- Soft Color : color as defined as configuration parameter MonitoringSoft_Limit_Violation (default is “yellow”).
- GWDU Color : color as selected as part of GWDU graph threshold attribute (default is green).

Mapping of monitoring result to foreground color of analog values is only applied if limits are defined for the enditems and the limit values were not modified by GWDU (color modification is permitted). All other values are displayed in GWDU Color.

When iconized, the overall monitoring status of the items referenced by the synoptic display is reflected by the icon color. The icon is drawn in

- normal foreground (as defined by desktop properties) color if the synoptic contains no data
- disabled color if monitoring of all items in the synoptic display is disabled
- “in limits” color if all items enabled for monitoring are in limit
- soft color if at least one item has a (delta) nominal limit violation and no item has a danger limit violation
- danger color if at least one item has a (delta) danger limit violation

For configuration of monitoring colors, see section8.3.2.5 Online Test Control Initialization File.

Acquisition Status/Processing Status

The acquisition and processing stati of the measurements are indicated by a flag in the upper left corner of the output objects (see figure Figure 8–71). If processing of at least one measurement is disabled, the acquisition flag is appended by a dash. If no dash is shown, all measurements displayed by the output object are enabled for processing.

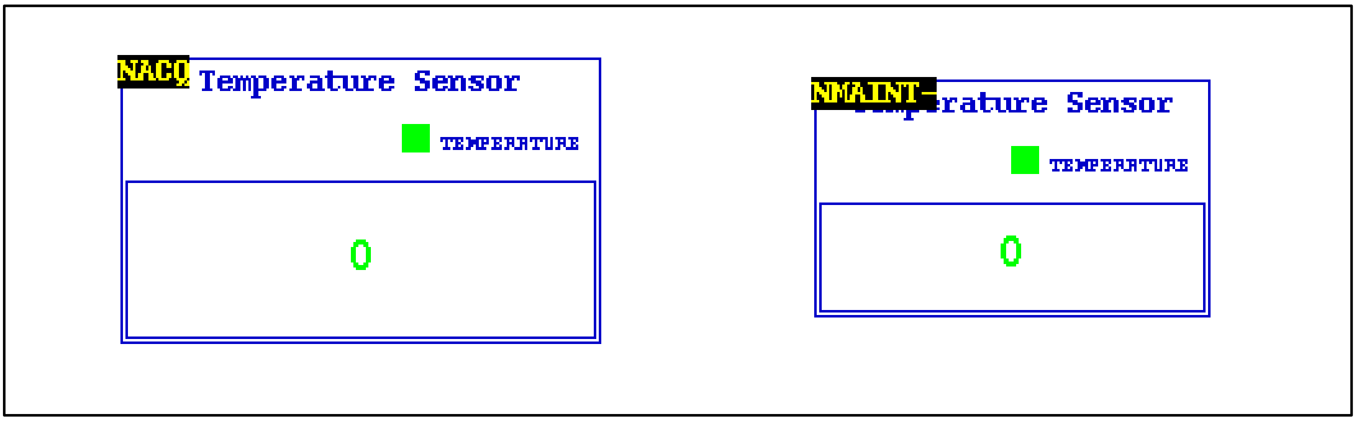


Figure 8–71 : *Data Not Acquired*

Flag	Description	Priority
INVAL	Invalid: the value is started for acquisition, but there was an error in the last delivery of the value.	0
NACQ	Not acquired: the value is not started for acquisition, i.e. no ADU is received for the value.	1
NRCD	Not received: the value is started for acquisition, but no ADU has been received yet.	2
NMAINT	Not maintained: the measurement is not maintained by any test node (e.g. not in test node item list, not referenced by an ADU).	3
REQUESTED	Requested: the value is requested from test node, but not yet delivered.	4
STATIC	Static: the latest value has been updated without any error, but the SAS has indicated a delivery problem.	5
INTERR	Interrupted: in the latest ADU there was an indication, that the delivery of the value has been interrupted before.	6
	Valid: the value is acquired and has been updated without any error.	7

Table 8–2 : *Acquisition Stati*

The priority is only applicable for output objects containing more than one measurements. In this case the acquisition status with the highest priority (0 = highest) is displayed. E.g. if one acquisition status is *NACQ* and the other is *STATIC* the *NACQ* flag will be displayed.

For configuration of acquisition/processing stati, see section8.3.2.5 Online Test Control Initialization File.

8.3.2.3.2.13.5 Synoptic Display Control Panel

The Synoptic Display Control Panel shows the overall monitoring status of all active synoptic displays. For each synoptic display a small status icon is drawn.

Figure 8–72 shows all possible stati (described from left to right):

1. The synoptic display contains at least one enditem with nominal (soft) limt violation, but none of the other have a danger limit violation.
2. Same as 1.
3. The synoptic display contains no enditem references (no data).
4. For all enditems in the synoptic, monitoring is disabled. The small status icon is highlighted because the mouse pointer is currenlty over this icon.
5. The synoptic display contains at least one enditem with danger limt violation.
6. All enditem enabled for monitoring are in limits.
7. The synoptic display is not in use (slot empty).
8. Same as 7.

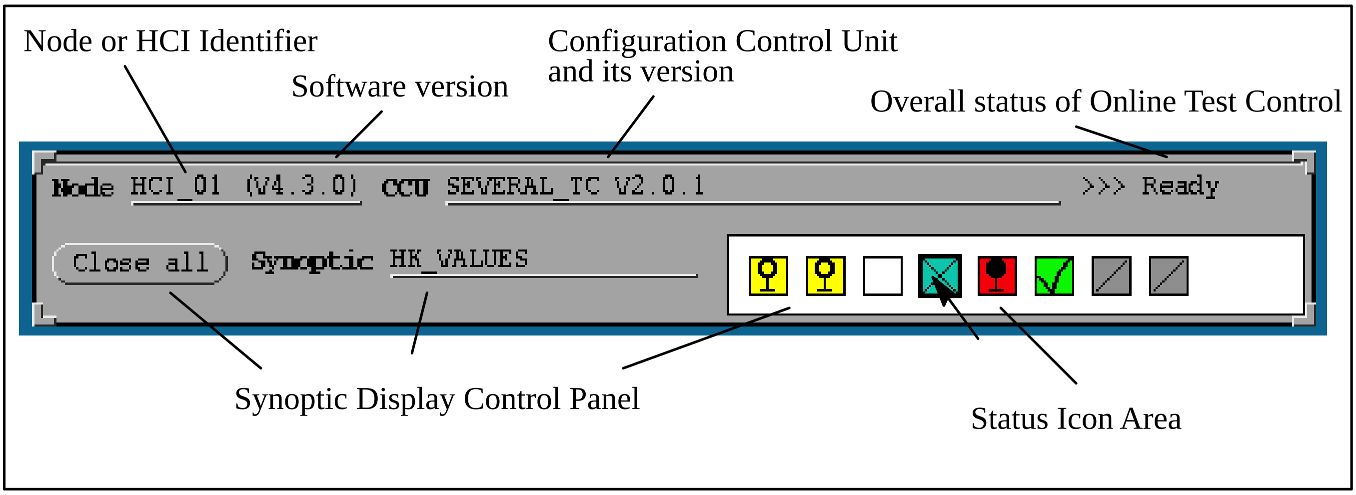


Figure 8–72 : *Synoptic Display Control Panel*

When starting with a screen setup saved with version 4.3 or older, the synoptic display control panel might be invisible. Resize the status display to get it visible.

When moving the mouse pointer over a small status icon it is highlighted and the last pathname element of the ground synoptic display is shown on the Synoptic field at the left side of the icon area. A left mouse button selection on a highlighted small status icon will open (maximize) the corresponding synoptic and bring it in front of all other windows (even if already open). A right mouse selection on a highlighted small status icon will close (minimize/iconize) the corresponding synoptic display. Selection of the “Close all” button will close (minimize/iconize) all synoptic displays on the screen.

Configuration

[STATUS_DISPLAY]
EmptyIconColor=gray
# small status icon color of an empty slot (unused synoptic)
NoDataIconColor=white
# small status icon color of a synoptic display without data

the monitoring colors are taken from the [COLOR] group (see hci.ini, section 8.3.2.5).

small icon image files (displayed acc. to monitoring status
of the synoptic display):

EmptySmallIcon=$HCI_HOME/config/empty.si
# synoptic display is not in use (empty)
No_DataSmallIcon=$HCI_HOME/config/no_data.si
# synoptic display has no data
In_LimitsSmallIcon=$HCI_HOME/config/in_limits.si
Soft_Limit_ViolationSmallIcon=$HCI_HOME/config/soft_limit_violation.si
Danger_Limit_ViolationSmallIcon=$HCI_HOME/config/danger_limit_violation.si
DisabledSmallIcon=$HCI_HOME/config/disabled.si
UndefinedSmallIcon=$HCI_HOME/config/undefined.si

8.3.2.3.2.13.6 Help on Synoptic Display

Synoptic Displays can provided detailed textual information about input and output objects. To get detailed information press the right mouse button when over an input/output object.

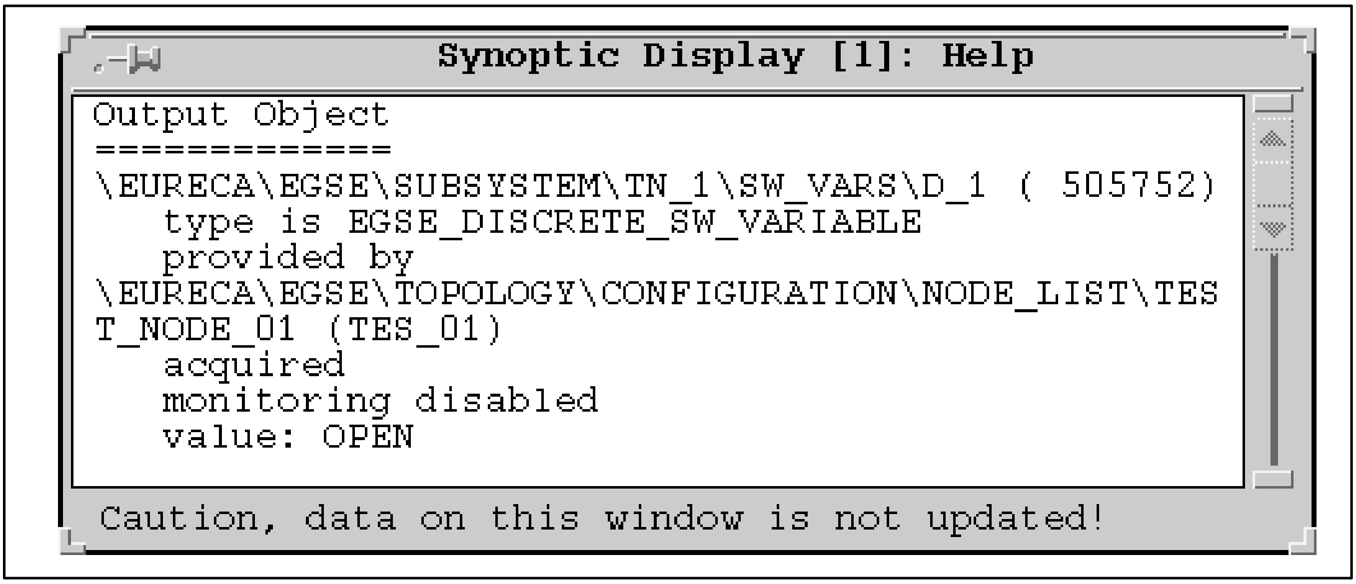


Figure 8–73 : *Synoptic Display: Help*

8.3.2.3.2.13.7 Termination of a Synoptic Display

The synoptic display can be terminated by selecting the menu item ”Quit” of the windows menu. The windows menu is selectable with right mouse click on the window title bar.

Configuration

For configuration of monitoring colors, flags,etc. refer to section 8.3.2.5 Online Test Control Configuration, group [SYNOPTIC_DISPLAY].

8.3.2.3.2.14 Input Dialog

The Input Dialog (see Figure 8–74) is invoked by the UCL library procedures READ_MESSAGE_FROM_USER and READ_NUMBER_FROM_USER or for critical command authorization.

The Input Dialog can be used to enter data to a running AP. The Input Dialog displays a prompt and what type of data is requested by that AP (in this case a string value). When the data was entered select either *Ok* to commit the input or *Cancel* to cancel the input.

When used for telecommand authorization, it prompts for password input. The characters type in are masked by *.

Typing <Return> key after data input has same meaning as selecting the Ok button.

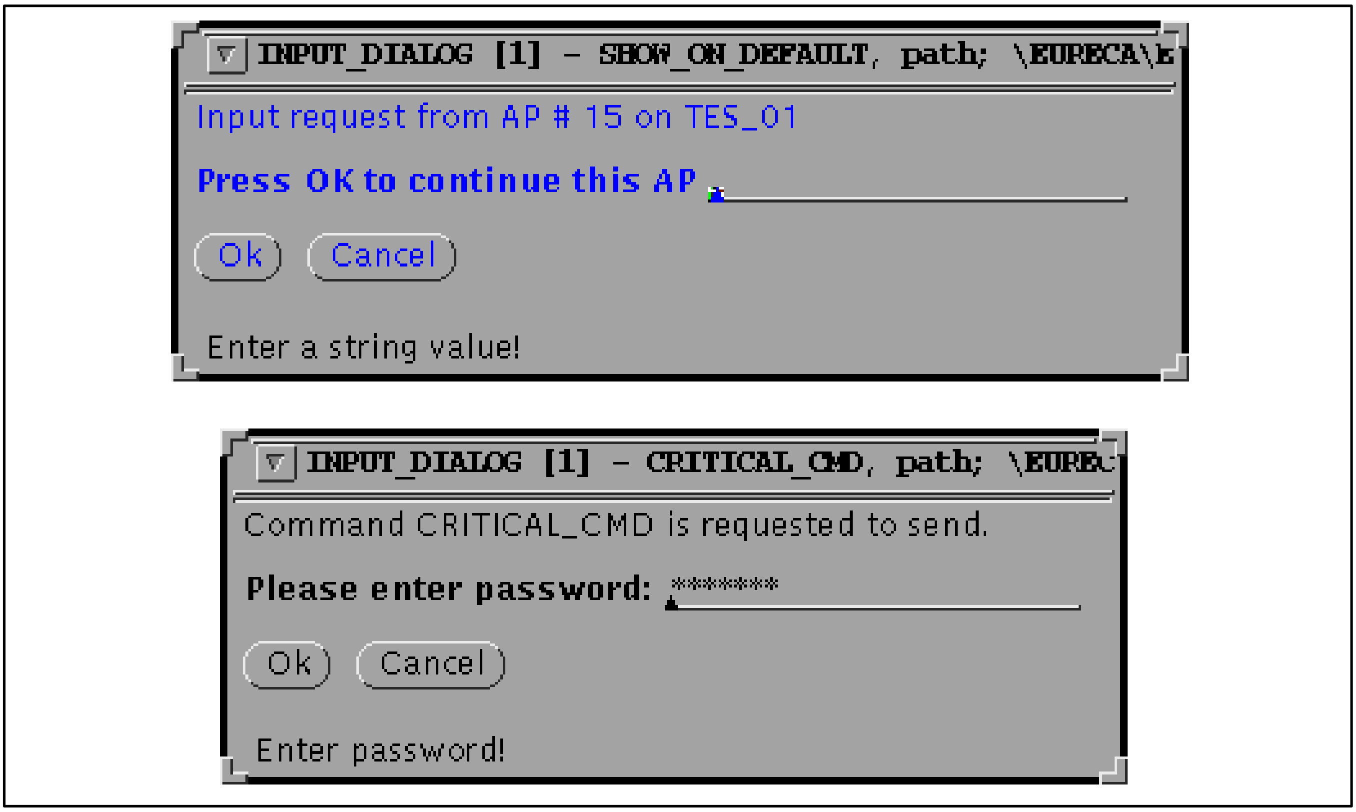


Figure 8–74 : *Input Dialog*

Sample for read message from user (as used for Figure 8–74):

variable AP_STATUS : UCL_RETURN;
variable MESSAGE : STRING(255);
...

```
READ_MESSAGE_FROM_USER
     (PROMPT          : ”Press OK to continue this AP”,
     WORKSTATION      : \\,
     USER_ENTRY       : MESSAGE,
     OPTIONS          : “–position 100 200 –foreground #0000FF“,
     STATUS           : AP_STATUS);
```

Legal options for Input Dialog are
–position x y
–size width height
–foreground color

Color can be specified by name (Unix command showrgb provides a list of names) or by RGB values (e.g. #0000ff).

8.3.2.3.3 Screen Setup Maintenance

To pop up the Screen Setup Maintenance dialog, select
Online_Test_Control –> Screen Setup Maintenance...

Screen setups are stored as ASCII files within the directory $GSAF_HOME/hci/data/screen_setup_pool. The content of this directory is displayed in ascending alphabetical order on the scrolling list of the Screen Setup Maintenance dialog (see Figure 8–75).

The following window applications can be part of a screen setup

d. AP Status (referred to as AP_STATUS)
e. Clock (referred to as CLOCK)
f. Command Facility (referred to as COMMAND_FACILITY)
g. Database Node Status (referred to as DB_NODE_STATUS)
h. Explorer (referred to as EXPLORER)
i. Go/Nogo Window (referred to as GONOGO_WINDOW)
j. Graph Facility (referred to as GRAPH_FACILITY)
k. Monitoring Window (referred to as MONITORING_WINDOW)
l. Online Test Control Menu (referred to as MAIN_MENU)
m. Raw Data Dump (referred to as RAW_DATA_DUMP)
n. Status Display (referred to as STATUS_DISPLAY)
o. Synoptic Display (referred to as SYNOPTIC_DISPLAY)
p. System Advisory (referred to as SYSTEM_ADVISORY)
q. SAS Status (referred to as SAS_STATUS)
r. Test Node Status (referred to as TEST_NODE_STATUS)

For each application one line defining the name of the application, window attributes, and window start parameter are given. The format is based on the command line arguments of XView. Comment lines beginning with a sharp # and empty lines can be inserted.

The following qualifier are supported:

- –position
  Sets the initial position of the application's base frame in pixels. The upper left corner of the screen is at position (0,0) with the x–axis increasing to the left and the y–axis increasing downward.
  Example: `CLOCK –position 100 200`
- –size
  Sets the width and height of the application's base frame. The values are in pixels.
  Example: `AP_STATUS –size 600 150`

- –node
  Sets the pathname of the default node of Command Facility.
  Sets the pathname of the test node for AP Status, SAS Status, and Test Node Status.
  Example: `TEST_NODE_STATUS –node \EURECA\EGSE\MTP`
- –default_path
  Sets the default path of Command Facility.
  Example: `COMMAND_FACILITY –default_path \EURECA\EGSE`
- –frequency
  Sets the update frequency in seconds of a Graph Facility.
  Example: `GRAPH_FACILITY –frequency 6`
- –graph
  Sets the graph used in a Graph Facility. The graph is specified by its number as defined in the configuration file (hci.ini). The default values are 1 (one) for bar chart, 2 (two) for line graph, 3 (three) for strip chart, and 4 (four) for text.
  Example: `GRAPH_FACILITY –graph 3`
- –login_sequence
  Executes the login sequence when starting a command facility. The login sequence can be located in database (\...) or file system (/...) depending on the separator.
  Example for database: COMMAND_FACILITY –LOGIN_SEQUENCE \SEQ\START
  Exampe for file system: COMMAND_FACILITY –LOGIN_SEQUENCE /seq/start
- –measurement
  Sets the pathname of the measurement used by the Graph Facility for animation.
  Example: `GRAPH_FACILITY –measurement \MOTOR\CURRENT`
- –measurement_list { list }
  Defines the measurements displayed in the Monitoring Window.
  Example: MONITORING_WINDOW –measurement_list { \MOTOR\CURRENT \MOTOR\TEMPERATURE }
- –minimum
  Sets the minimum value of the value axis for a Graph Facility.
  Example: `GRAPH_FACILITY –minimum –280`
- –maximum
  Sets the maximum value of the value axis for a Graph Facility.
  Example: `GRAPH_FACILITY –maximum 1E+02`
- –picture
  Sets the pathname of the picture load by Synoptic Display.
  Example: `SYNOPTIC_DISPLAY –picture \EURECA\EGSE\WELCOME`
- –replay
  Sets the Clock to replay mode.
  Example: CLOCK –REPLAY

Pathnames must be written in **upper case** letters.

The following is an example of a screen setup file:

```
#CCU:PERF1
CLOCK –POSITION    2  141 –SIZE  266   63
MAIN_MENU –POSITION    6   97 –SIZE  179   44
STATUS_DISPLAY –POSITION    5  863 –SIZE 1142   32
```

```
SYNOPTIC_DISPLAY –POSITION  649  394 –SIZE  491  255 –PICTURE \EURECA\EGSE\SUBSYSTEM\CONTROL\SWITCHES
SYNOPTIC_DISPLAY –POSITION  189    0 –SIZE  949  726 –PICTURE \EURECA\EGSE\SUBSYSTEM\CONTROL\OVERVIEW
SYNOPTIC_DISPLAY –POSITION  902  699 –SIZE  240  105 –PICTURE \EURECA\EGSE\SUBSYSTEM\CONTROL\MODE_PICTURE
```

The screen setup maintenance facility allows to maintain such files by saving the actual screen layout to the file. It allows to delete such a file, to load a new screen layout by specifying the name of the setup file or to rename such a file. The names of the existing files are displayed in scrollable list. The operations Load, Delete and rename are executed on the selected name.

Load Screen Setup

To load a screen setup select a screen setup name from the list and press the *Load* button. This will pop up a confirmation box as shown in Figure 8–37. After confirmation the selected screen setup is loaded. The current windows are all deleted from the screen and replaced by those defined in the selected screen setup.

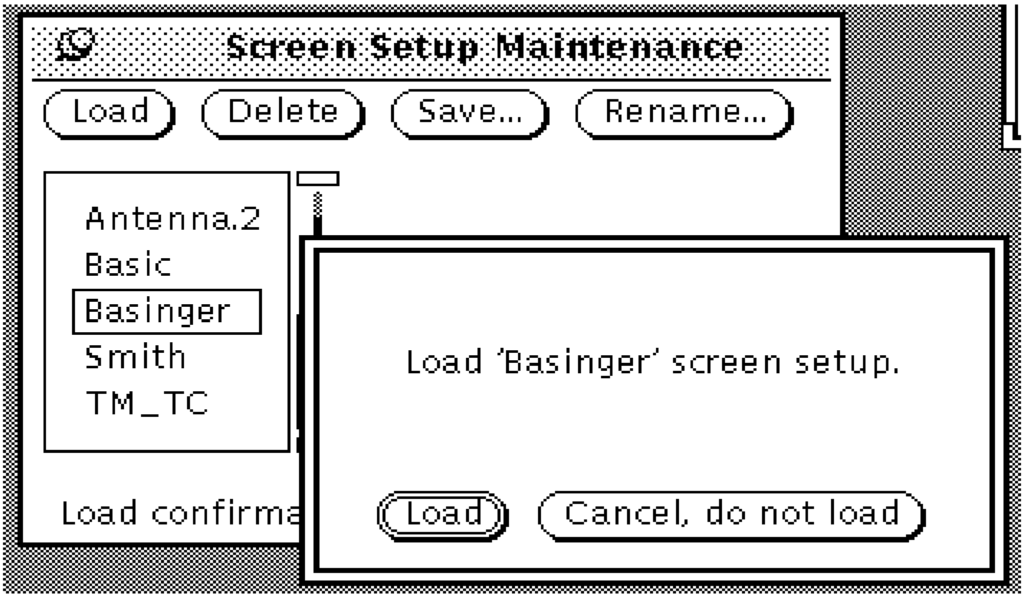


Figure 8–75 : *Load Screen Setup*

If the screen setup was saved in a different configuration, a warning is included in the confirmation box (see Figure 8–76). To suppress the configuration check, e.g. when using a basic setup containing no configuration dependant information, edit the screen setup file (e.g. using the command ”vi $HCI_HOME/data/screen_setup_pool/basic”) and remove the line containing the CCU identification (e.g. #CCU:PERF1).

In general, it is not recommended to load screen setups from different configuration. Some applications may not be started because database references can be solved.

Figure 8–76 : *Load Screen Setup (Different Configuration)*

<u>Delete Screen Setup</u>

To delete a screen setup select a screen setup name from the list and press the *Delete* button. After confirmation the selected screen setup is deleted from the setup pool.

Note: Do not delete screen setups that are referenced in EGSE_USER_PROFILE enditems of the Mission Database (MDB) without deleting that reference too.

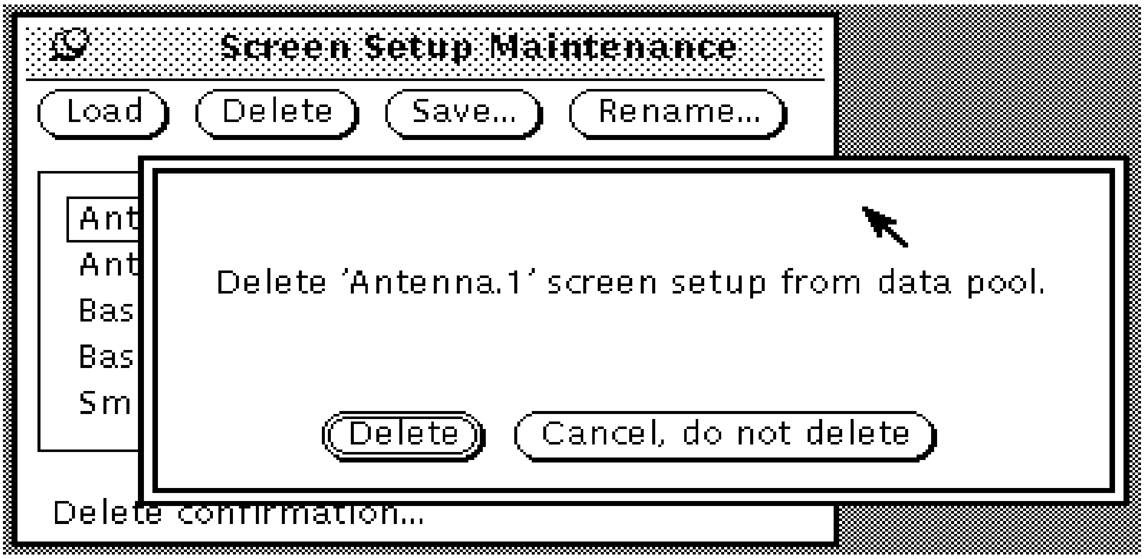


Figure 8–77 : *Delete Screen Setup*

Save Screen Setup

To save the actual screen layout press the *Save...* button. This will pop up a Save Screen Setup dialog on which the name of the new screen setup must be entered. The names are case sensitive, ”demo” and ”Demo” are not the same. After selection of the *Save* button on the Save Screen Setup dialog the current Online Test Control applications are scanned and the new screen setup definition file is stored in the setup pool. The saved screen setup may be called up later again using the Load operation or by referencing the name in the user’s profile within the MDB which will Online Test Control force to load the screen setup automatically when started again.

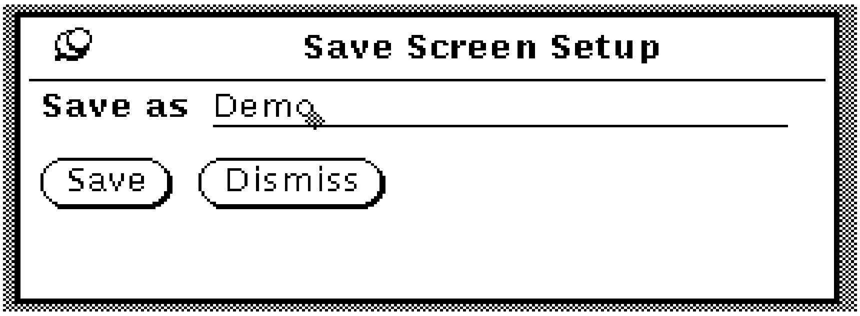


Figure 8–78 : *Save Screen Setup Dialog*

Rename Screen Setup

To rename a screen setup select the corresponding item on the list and press the *Rename...* button. You can then enter on the Rename Screen Setup dialog the new name of the screen setup. After selection of the *Rename* button of the dialog the new name will apply. and the selected screen setup is renamed in the setup pool.

Note: Do not rename screen definitions that are referenced in EGSE_USER_PROFILE enditems of the MDB without renaming that reference too.

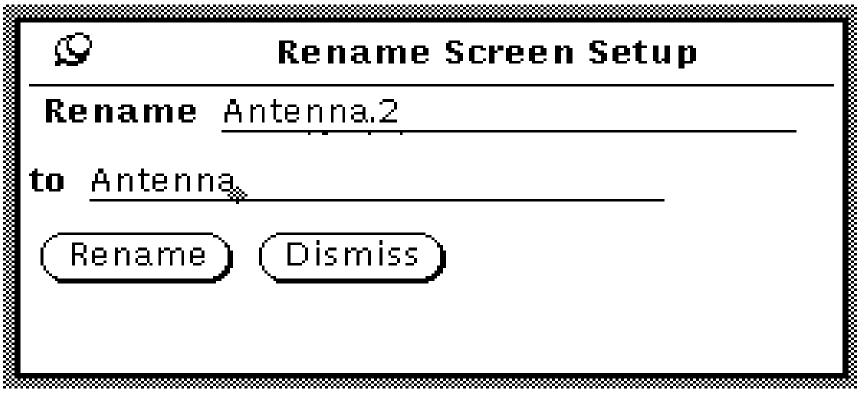


Figure 8–79 : *Rename Screen Setup Dialog*

8.3.2.3.4 User Services

To pop up the User Services menu, select
Online_Test_Control –> General *Services* –> *User Services*

The User Services are only available if private screen services have been defined in the user's profile (refer to I_MDB Navigator). To start a user service select the corresponding item from the User Services menu. The corresponding command will then be started in batch mode. Errors and status information about start and termination of the command are routed to the CGS Message Window.

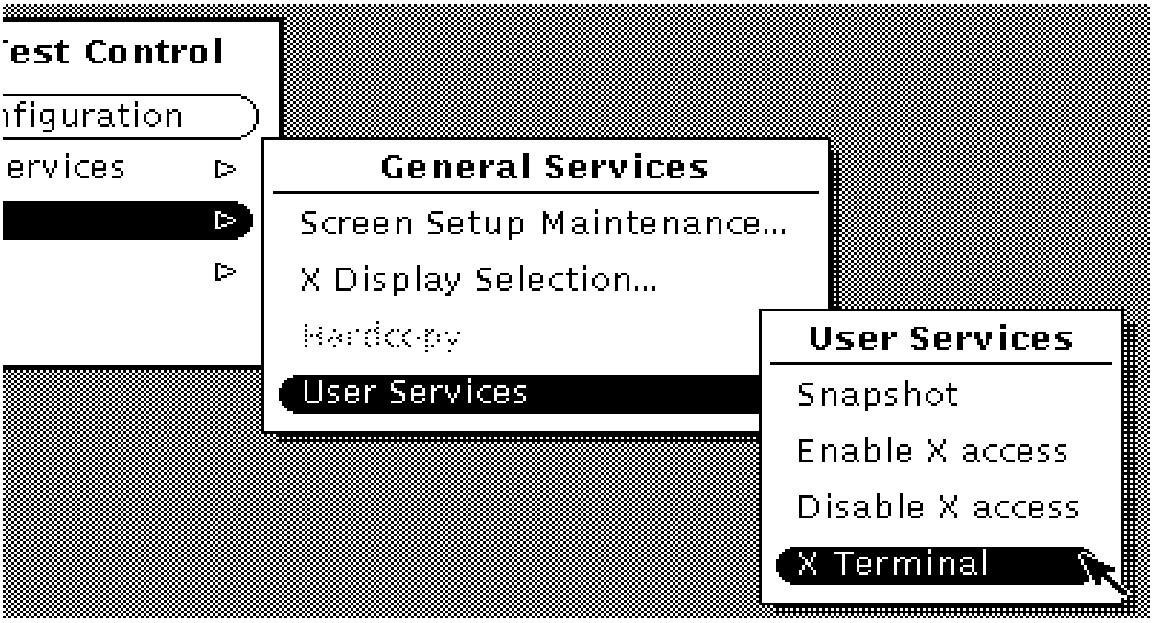


Figure 8–80 : *User Services Menu*

8.3.2.3.5 Online Help

To get a short online help information move the cursor anywhere on a Online Test Control window and press the <Help> key on the keyboard. In such cases a help window is displayed providing information about that Online Test Control window.

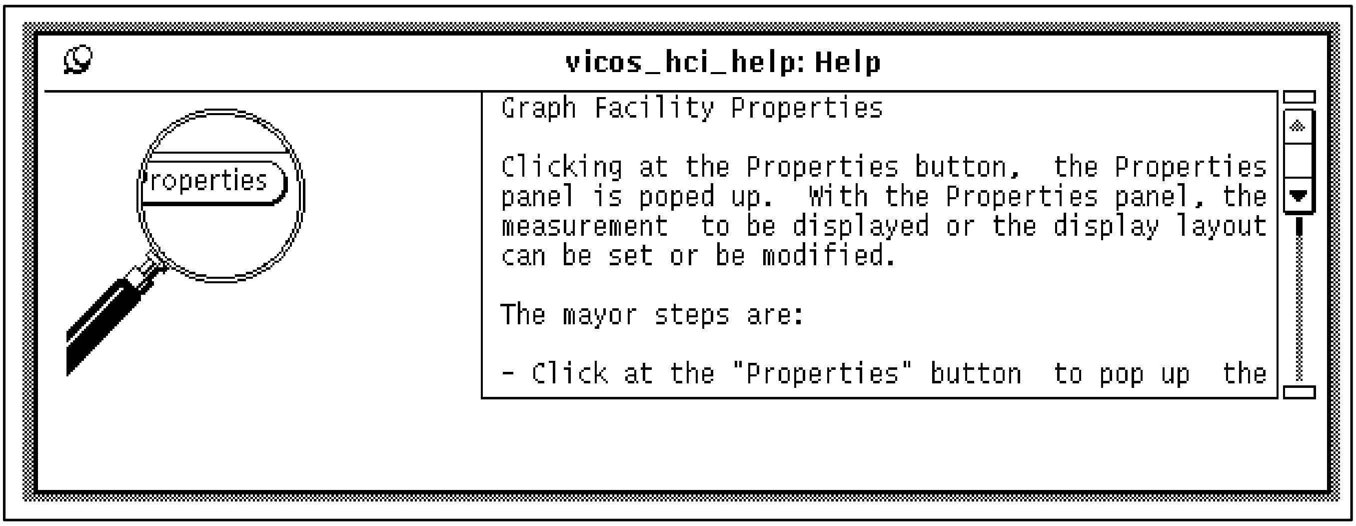


Figure 8–81 : *Online Help Window*

8.3.2.3.6 Exit

To exit online test control, select
Online_Test_Control –> Exit

This will pop up a confirmation window as shown in Figure 8–82. Exiting Online Test Control will terminate all its applications and services that have been started, but it will not terminate online help manual viewers or user services.

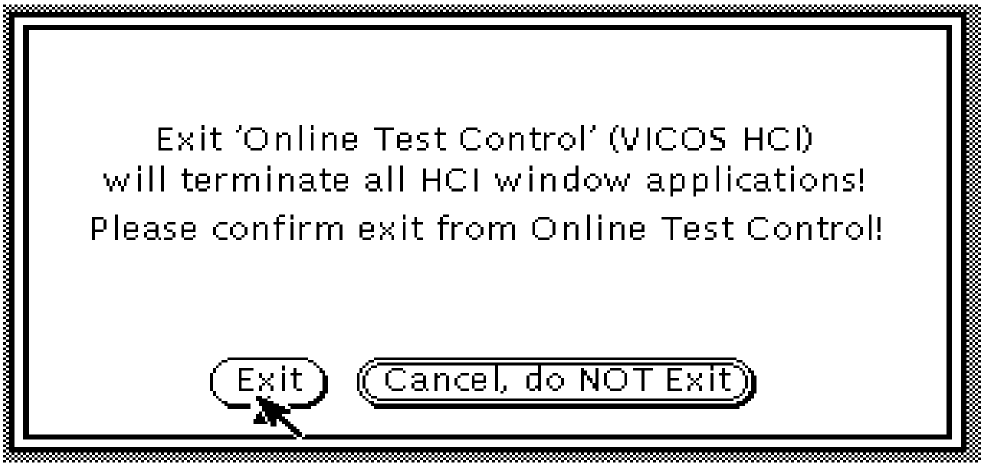


Figure 8–82 : *Exit Online Test Control*

8.3.2.4 Online Test Control Icons

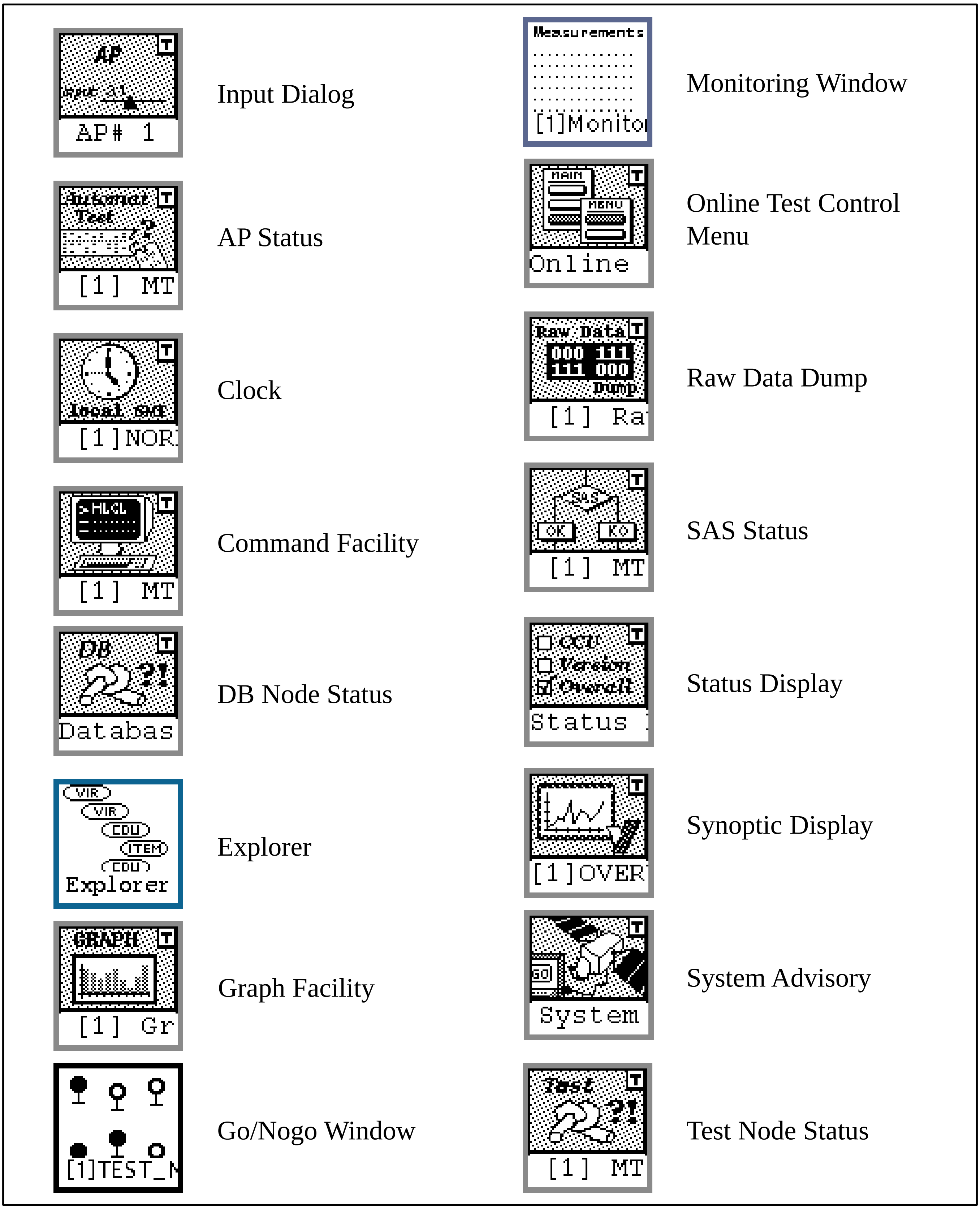


Figure 8–83 : *Online Test Control Icons*

8.3.2.5 Online Test Control (HCI) Initialization File (hci.ini)

```
#####################################################################################
# Online Test Control Configuration
#
# Version: @(#) hci.ini /main/cgs_4.1.1patches/cgs_4.2/cgs_4.3/cgs_4.4/2  09/26/00 10:35:27@(#)
#
# General Description
#
# The Online Test Control initialization file hci.ini is used to configure the
# Online Test Control software at initialization time. When started the file
# hci.ini located at $HCI_HOME/config is read and Online Test Control is
# initialized according to the definitions in hci.ini.
#
# The hci.ini file is separated into groups which may have attributes. A group
# is indicated by a name enclosed by square brackets, an attribute definition
# consists of the attributes name followed by the assignment operator = and
# the attributes value. Group and attribute names are case insensitive.
# An attribute definition is terminated by the end of line,
# i.e. it is not allowed to continue a definition on the next line. Comments
# may be inserted everywhere in the hci.ini file, but only on separate lines.
# A comment line must always start with a sharp # and is terminated by the end
# of line.
#
# Groups and Attributes
#
# General and overall attributes of Online Test Control are defined in the
# group ONLINE_TEST_CONTROL.
#####################################################################################


[ ONLINE_TEST_CONTROL ]

CommunicationRequired = True
# Defines whether Online Test Control requires OK status when setting up the
# communication software. If True, it will abort execution on setup errors.

DataRequestRetryDelay = 60.0
# If data can’t be requested (e.g. for the AP Status) Online Test Control
# will retry the request after a.m. delay.
# Default is every 60 seconds.

DataRegistrationTimeOut = 60
# Defines the time in seconds HCI will wait for the test node until
# it registers its data (data distribution table).
# Default is 60 seconds.

EndItemTimeOut = 20.0
```

```
# Minimum number of seconds Online Test Control will wait until
# an end item value is delivered from a test node (HLCL commanding).

ExitTimeOut = 45
# Time out after which Online Test Control will ask for a forced shut down.
# Default is 45 sec. (60 sec. in CGS 3.1.2)

MaxAPCommands=10
# Maximum number of AP commands (e.g. start/abort AP, execute library routine)
# that can be executed in parallel.

MaxDeliveryNotes = 500
# Maximum number of delivery notes (data requests and cancel notes)
# that can be handled.
# Default is 500.

MaxEndItemRequests=10
# Maximum number of enditems that can be queried by HLCL commands
# at the same time.
# Default is 10.

MaxTestNodes = 4
# Maximum number of test nodes supported by Online Test Control.
# Default is 4.

MaxWindowsPerApplication = 32
# Maximum number of window instantiations of each kind of application
# (e.g. command facility, AP status display). A MaxWindow attribute
# (see window application groups, e.g. MaxWindow of [ AP_STATUS ] must
# never exceed this constant.
# Default is 32.

NonInterruptableDBAccess = False
# Defines the method of synchronization of mission database access.
# Default is false, i.e. it is interruptable.

#####################################################################################
# Attributes, that should never be modified:
#####################################################################################

APClientTimeOut=2.0
# Internal configuration item, should never be modified, default is 2.0

HomeDirectory = /gsaf/hci
# Defines the home directory, default is $HCI_HOME; must always be omitted.

ConfigDirectory = /gsaf/hci/config
# Defines the data directory, default is $HCI_HOME/config;
```

```
# must always be omitted.

DataDirectory = /gsaf/hci/data
# defines the data directory, default is $HCI_HOME/data; must always be omitted.

IconDirectory = /gsaf/hci/config
# Defines the icon directory, default is $HCI_HOME/config;
# can be used to define own icons for Online Test Control applications.

DataViewsSearchPath = ”. /tools/DataViews /tools/DataViews/etc”
# Defines the DataViews search path, default is $GSAF_HOME/gwdu/lib/layouts,
# $GSAF_HOME/gwdu/lib/pred_elements, $DV_HOME, $DV_HOME/etc, $DV_HOME/lib
# $DV_HOME/lib/fonts, $DV_HOME/lib/icons, $DV_HOME/lib/images,
# $DV_HOME/lib/templates, $DV_HOME/lib/views”;
# must always be omitted.

MPS_Directory = /gsaf_home/mps
# Defines the MPS home directory; must always be omitted

VersionIdTable=/gsaf/hci/data/version_id_table.dat
# Defines the location of the version id table, default is
# $GSAF_HOME/cgs/config/version_id_table.dat

DelayTask = false
# Adds a CPU consumer task to avoid blocking of HCI, default is false.

ReceiverTaskSize = 200_000
# HCI message receiver task size in bytes; default is 200_000.

Debug = Off
# Produces debug output when reading hci.ini, default is off

EnditemListSize=4_000
# Defines the number of enditems that can be stored in a message
# of type ENDITEMS_MESSAGE/APPEND_ENDITEMS (data distribution table/
# announce enditems).

#####################################################################################
# General Window Attributes
#####################################################################################

[ WINDOWS ]

PathnameStoredLength = 200
# Defines the maximum number of character for pathname in input
# text fields (e.g. name of the measurement in the Graph Facility
# Properties dialog); default is 200.
```

```
PathnameDisplayLength = 200
# Defines the maximum number of character for pathname displayed
# in input text fields; default is 200. If the pathname exceeds the
# display length scroll–buttons are added to the text field.

[COLOR]
# Decription of Color Specifications
# Colors can be specified by using color name defined in the color
# database (use Unix "showrgb" to look up color database) or
# hexadecimal code. A hexadecimal code is specified as an initial sharp
# sign character followed by a hexadecimal specification in one of
# the following formats:
#       #RGB            (one character per color)
#       #RRGGBB         (two character per color)
#       #RRRGGGBBB      (three character per color)
#       #RRRRGGGGBBBB   (four character per color)
# where R, G, and B represent single haxadecimal digits (upper or
# lower case).
# When fewer than 16 bits each are specified, they represent the most
# significant bits of the value, For example, #3a7 is the same
# as #3000a0007000.


# Definition of colors displayed acc. to monitoring result
MonitoringDisabled=turquoise
MonitoringIn_Limits=green
MonitoringSoft_Limit_Violation=yellow
MonitoringDanger_Limit_Violation=red
MonitoringUndefined=#4ED

#################################################################################
# Window Application Groups
#################################################################################


[ AP_STATUS ]

MaxWindows = 2
# Defines the maximum number of AP Status Displays that can be displayed in
# parallel.
# Default is 4.

UpdateRate = 5
# Defines how often the AP status window is updated (in seconds); the value 0
# means it will be updated only if one or more of the values are changed.
# Default is 5.
```

```
[ CLOCK ]

MaxWindows = 4
# Defines the maximum number of Clock windows that can be displayed in parallel.
# Default is 4.

ReplayFooter=Yes
# Controls display of window footer when started in replay mode, default
# is footer is displayed.

UpdateRate = 5
# Defines how often the Clock window (replay mode only) is
# updated (in seconds); the value 0 means it will be updated only if one
# or more of the values are changed.
# Default is 0.

SMT_Refresh = 0.333
# Refresh rate in seconds of SMT reading.
# Default is 0.333 seconds, i.e. the SMT is read every 1/3 second.

# Time Code Identifier
LocalTimeTCI=LT
SMT_TCI=SMT

[ COMMAND_FACILITY ]

MaxWindows = 4
# Defines the maximum number of Command Facilities that can be displayed in
# parallel.
# Default is 4.

History = 200
# Defines the size of the history buffer, i.e. the number of commands
# stored in the history.
# Default is 200.

DontLog=? LIST
# Specifies a list of commands not to be logged (this is applicable
# for HLCL command windows, but not synoptic displays).
# The list may include:
#        – a command/procedure name for primary commands,
#        – a reserved word (import, type) for commands that start
#          with a reserved word,
#        – a pathname for APs and command sequences from MDB,
#        – a qualified name (pathname.identifier) for UCL library
#          procedures,
#        – a file name in string quotes for command sequences from
#          files,
```

```
#          – ”?” for the ? command,
#          – ”:=” for assignments.
#          – ”*” for all commands (to disable logging)
#
#  LOG_SYNTAX_ERRORS en/disables logging for syntactically wrong commands.
#  If logging is disabled with DONT_LOG = ”*” nothing will be logged
#  even if LOG_SYNTAX_ERRORS is set.


[ EXPLORER ]

ButtonWidth = 128
# Pixel width of the buttons.

ButtonXOffset = 64
# Pixel offset for a new tree level related to its parent level.

ButtonYOffset = 1
# Pixel distance of buttons in y direction.


[ GONOGO_WINDOW ]
MaxWindows = 4
# Defines the maximum number of Go/Nogo Windows that can be displayed in
# parallel, default is 4.


[ GRAPH_FACILITY ]

Default = \EURECA\EGSE\SUBSYSTEMS\ANT\MEASUREMENTS\...
# Sets a default measurement name used when a Graph Facility comes up,
# default is not set. This attribute should only be used if the pathname
# is used very often.

MaxWindows = 4
# Defines the maximum number of Graph Facilities that can be displayed in
# parallel, default is 1.

GraphName1 = ”Bar Chart”
GraphTemplate1 = ”bar_chart.dv”
GraphName2 = ”Line Graph”
GraphTemplate2 = ”line_graph.dv”
GraphName3 = ”Strip Chart”
GraphTemplate3 = ”strip_chart.dv”
GraphName4 = ”Text”
GraphTemplate4 = ”text.dv”
# The attributes GraphNameN and GraphTemplateN define the graphs that can be
```

used by the Graph Facility. In general, these attributes should not be
modified.

[INPUT_DIALOG]
MaxWindows = 10
# Maximum number of input dialog windows that can be displayed in parallel;
# default is 10.
InputFieldDisplayLength = 20
# Display length of the input field in character.

[MAIN_MENU]

XDisplayDialog = Off
No longer supported.
Toggles the X Display Selection, default is Off.

EnableDisconnectedNodes = Off
# If on disconnected nodes are selectable on the test node submenu,
# default is Off.

[MONITORING_WINDOW]
MaxWindows = 4
# Defines the maximum number of Monitoring Windows that can be displayed in
# parallel, default is 4.

[RAW_DATA_DUMP]

MaxWindows = 1
# Maximum number of raw data dump tools that can be displayed in parallel;
# default is 1.

PacketDefault = \None
# Default packet pathname, displayed when the properties are poped up.
# Should be omitted.

The following attributes define the layout and behaviour of the properties
dialog:

BaseDefault = 16
# Defines the base–button selected by default (button with base 16).

BaseCount=5
# Defines the number of base–buttons; default is 3 (default bases are
# 16, 10, ASCII).

```
Base1=2
Base2=8
Base3=10
Base4=16
Base5=128
# BaseN define the base choosen after selection of base–button N, e.g. after
# selection of base–button 2 the output is displayed in octal format. A base
# between 128 and 255 is interpreted as ASCII.

BytesPerLineDefault = 32
# Bytes–per–line–button selected by default.

BytesPerLineCount=8
# Number of bytes–per–line–buttons selected by default; default is 8.

BytesPerLine1=8
BytesPerLine2=16
BytesPerLine3=24
BytesPerLine4=32
BytesPerLine5=40
BytesPerLine6=48
BytesPerLine7=56
BytesPerLine8=64
# BytesPerLineN define the bytes per line choosen after selection of
# bytes–per–line–button N, e.g. selection of the third button will
# display 24 lines. Defaults are N multiplied with eight.


[ SAS_STATUS ]

MaxWindows = 4
# Defines the maximum number of SAS Status Displays that can be displayed in
# parallel, default is 4.

UpdateRate = 0
# Defines how often the AP status window is updated (in seconds); the value 0
# means it will be updated only if one or more of the values are changed.
# Default is 0.


[ SYNOPTIC_DISPLAY ]
# When RGB colors are modified, colors defined in the DataViews
# color lookup table should be used. Otherwise colors may be displayed
# different from the RGB values.
# Default color lookup table: $DVHOME/etc/default.clut

MaxWindows = 8
# Defines the maximum number of Synoptic Displays that can be displayed in
```

parallel, default is 8.

MinimumWindowSize=50
# Minimum window size accepted for synoptic displays, default is 50.

MaxActions=500
# Maximum number of actions (HLCL commands, menu, picture replacer) in
# one synoptic. Should not be modified.

MaxMenuItems=10
# Maximum number of menu items of a popup menu in synoptics.

MaxVariables=50
# Maximum number of variables that can be displayed in one synoptic.
# Should not be modified.

MaxPictureReplacer=4
# Maximum number of picture replacements in parallel. Should not be modified.

DisplayNotMaintainedItems = Disable
# If enabled all items of a synoptic display that are not maintained by
# any test node are reported to the message handler; default is Disable.

NACQFlagBackgroundRed = 0
NACQFlagBackgroundGreen = 0
NACQFlagBackgroundBlue = 0
# Define the background color of the ’data acquisition’ flag as RGB value,
# default is black.

NACQFlagForegroundRed = 255
NACQFlagForegroundGreen = 255
NACQFlagForegroundBlue = 0
# Define the foreground color of the ’data acquisition’ flag as RGB value,
# default is yellow.

NACQFlagTextSize = 2
# Defines the text size of the ’data not acquired’ flag, 1 is very small,
# 9 very large. Default is 2 (small).

# Text displayed on acquisition status flag
NOT_MAINTAINEDFlagText = NMAINT
REQUESTEDFlagText = REQUESTED
NOT_ACQUIREDFlagText = NACQ
NOT_RECEIVEDFlagText = NRCD
INVALIDFlagText = INVAL
ACQUIREDFlagText = ””
DATA_INTERRUPTIONFlagText = INTERR

```
STATICFlagText = STATIC

# If an output object has more than one measurement connected
# to it (e.g. a bar chart displaying \MEA_1 and \MEA_2)
# Synoptic Displays uses a priority list to determine the
# flag text, 0 is highest priority, 7 lowest (i.e. if \MEA_1
# is NOT_ACQUIRED and \MEA_2 is STATIC, NOT_ACQUIRED will be
# displayed.
NOT_MAINTAINEDPriority = 3
REQUESTEDPriority = 4
NOT_ACQUIREDPriority = 1
NOT_RECEIVEDPriority = 2
INVALIDPriority = 0
ACQUIREDPriority = 7
DATA_INTERRUPTIONPriority = 6
STATICPriority = 5

Stretching = False
# If enabled, stretching makes the portion of the synoptic containing
# objects (graphs, input objects, static elements, etc.) exactly fit in
# the window. Stretching transforms the object’s control points differently
# in the x and y dimensions.
# For that reason, stretching is automatically disabled for synoptics
# containing arcs and circles. default is false.

SameAspectRatio = False
# If disabled, the portion of the synoptic containing objects fit in the
# window using a best fit algorithm to preserve the aspect ratio.
# In this case, either the top and bottom or the sides of the synoptic
# may not be visible. If enabled, the synoptic is drawn exactly as it
# appeared in GWDU. Default is false.

RedrawAreas = no
# If enabled, all dynamic object areas are redrawn after update to show
# static objects on top of these. Such static objects (e.g. a line crossing
# a bar chart) are hidden by the update procedure.
# Default is disabled for performance reasons.

HelpWindowWidth = 450
HelpWindowHeight = 120
# Default window width and size of the synoptic display help.

HelpWindowMemoryMaximum = 5_000
# Memory used by help window, should not be changed.

RestoreOriginalColorWhenNotMonitored = no
# Restores original color of output element when monitoring is disabled.
# If false, Disabled* color is used.
```

RestoreOriginalLimitColorsWhenNotMonitored = no
# Restores original limit colors of output element when monitoring is disabled.
# If false, Disabled* color is used.

[STATUS_DISPLAY]

CCUItemWidth = 40
# Defines the size of the CCU item in character, default is 40.

CCUItemNextRow = Off
# Defines if the CCU item shall be displayed on next row, default is off.

CCUItem = On
# Defines whether the CCU item shall be shown

attributes for synoptic display's small icons:

EmptyIconColor=gray
# color displayed if synoptic display slot is unused, i.e. no
# synoptic display is visible

NoDataIconColor=white
# color displayed if corresponding synoptic display contains no data

small icon image files (displayed acc. to monitoring status
of the synoptic display):

EmptySmallIcon=$HCI_HOME/config/empty.si
# synoptic display is not in use (empty)
No_DataSmallIcon=$HCI_HOME/config/no_data.si
# synoptic display has no data
In_LimitsSmallIcon=$HCI_HOME/config/in_limits.si
Soft_Limit_ViolationSmallIcon=$HCI_HOME/config/soft_limit_violation.si
Danger_Limit_ViolationSmallIcon=$HCI_HOME/config/danger_limit_violation.si
DisabledSmallIcon=$HCI_HOME/config/disabled.si
UndefinedSmallIcon=$HCI_HOME/config/undefined.si

[SYSTEM_ADVISORY]

UpdateFrequency = 5
# Defines how often the System Advisory shall be updated in seconds,
# default is 5.

CheckFrequency = 15

```
# Defines how often it shall be checked that the System Advisory was updated,
# default is two times the update frequency.

AcknowledgeService = On
# Determines if the acknowledge service is toggled on or off; default is on.

AcknowledgeTimeOut = 10
# Sets when the acknowledge time out in seconds, i.e. the time after that
# Online Test Control will log that the alarm was not acknowledged.

BeepOnWarning = true
# Enables beep on warnings (yellow color); default is true.


[ TEST_NODE_STATUS ]

MaxWindows = 4
# Defines the maximum number of Test Node Status Displays that can be
# displayed in parallel, default is 4.

AutoResize = True
# Defines whether test node status automatically resizes the window
# after selection of a new group; default is true.


##################################################################################
# Any other groups
##################################################################################


[HCI_RPI]

# Provides attributes to configure the communication between
# Online Test Control and TES

LoadSynopticTimeOut=20_000
# Defines the time in milliseconds TES will wait until the load_synoptic
# library call is timed out, default is 20_000.

Last_HCI_ApplicationId=32
# Highest identifier that can be used for a HCI application,
# default is 32 (i.e. HCI_32).

ReceiverTaskSize = 200_000
# Size of the receiver task in bytes, default is 200_000. Must not be modified.

PollingInterval=333
# Polling intervall (milliseconds) for network software message reception,
```

default is 333. Must not be modified.

[HELP]

Online = vicos_hci_help
# Defines the name of the online help file, default is vicos_hci_help.
# Must not be modified.

Path = /gsaf/hci/config
# Defines the path where the online help files are located,
# default $HCI_HOME/config.
# Must not be modified.

[HLCL]

SequenceDirectoryVariable = ”HOME”
# Defines the location where to find the login/logout sequences,
# default is the user’s home directory.

LoginSequenceFile = ”/.user/hlcl_login.seq”
# Defines the rest of the login/logout sequence file names,
# e.g. $HOME/.user/hlcl_login.seq.

[LOG]

LoadErrorMessages = True
# Controls loading of error message definitions. If disabled error
# messages will neither be loaded from file nor from data base.
# Default is true.

LoadErrorMessagesFromDB = True
# Controls loading error message definitions from data base.
# If disabled error messages are only loaded from file.

BufferSize = 100
# Defines the size of Online Test Control’s internal log buffer,
# default is 100, should not be modified.

Required = True
# Defines whether Online Test Control requires logging services,
# i.e. if required and a connection to the TRDB is not possible
# it will terminate, default is True.

Disabled = false
# Disables logging totally.

```
CodeOffset = 5000
# Offset added to the internal error codes of Online Test Control.
# Should not be modified.

MessageFile=/gsaf_home/hci/config/messages.def
# Defines the location of the message definition file.
# Default is $GSAF_HOME/hci/config/messages.def

UserName=””
# Defines the user name of the addressed message window, default is
# all users (”” is same user as Online Test Control, ”*” is all users).


[ SCREEN_SETUP ]

ConfigurationCheck = On
# Defines whether a warning/confirmation is generated when a screen
# setup has to be loaded that was stored under a different configuration.
# Default is yes.

LineWidth = 256
# Defines the maximum line width/length in a screen setup file.
# Default is 256.

ListRows = 5
# Defines the number of visible rows of the screen setup files list,
# default is 5.

Directory = /gsaf/hci/data/screen_setup_pool
# Defines the location of the screen setup pool directory,
# default is $HCI_HOME/data/screen_setup_pool.


# [ SMT_SIM ]
#
# Defines attributes for the SMT Simulator
#
# Offset = 0.0
# Offset to local time in seconds.
#
# Available = true
# Enables/disables SMT Simulator


[ TASKING ]

# Defines the internal attributes for tasking, must NOT be modified/set.
```

INPUT_DIALOG.T_INPUT_SENDER = 128_000
INPUT_DIALOG.T_INPUT_OBCS = 64_000
APPLICATION_CONTROLLER.INTERIM_BUFFER_TASK_SIZE = 20_000
COMMAND_DISTRIBUTOR.T_OBCS_DELIVERY_NOTE = 10_000
COMMAND_DISTRIBUTOR.T_SERVER=100_000
CMD_WINDOW_MANAGER = 300_000
DATA_DISTRIBUTOR.INTERIM_BUFFER_TASK_SIZE = 48_000
EXPLORER.T_CREATE_TASK = 50_000
HCI_ENVIRONMENT = 32_000
HCI_SMT.T_SMT_READER = 32_000
LOGGING_SERVICE.INTERIM_BUFFER_TASK_SIZE = 48_000
ScreenSetupLoader = 10_000
SD_ANSWER_DIALOG.T_ANSWER_TASK = 10_000
SYNOPTIC_DISPLAYS_EXECUTOR.T_EXECUTOR = 130_000
SYNOPTIC_DISPLAYS_EXECUTOR.T_PICTURE_REPLACER = 130_000
T_OBCS_DATA_DISTRIBUTOR = 200_000
T_HLCL_INTERPRETER = 46_000
T_INTERPOSER_TASK = 300_000

[VALIDATION]

Controls the DataViews validation (license test).

Validation = On
# Toggles validation on/off, default is On.

Show = On
# Controls display of the validation window, default is window is shown (on).

WindowWidth = 400
WindowHeight = 400
# Defines the size of the validation window, default is 400, 400.

ErrorDiversion = On
# Enables diversion of DataViews error messages to synoptic displays,
# default is on. If off, error messages are displayed on standard error
# (i.g. the console window).

Logo = $GSAF_HOME/hci/config/logo.v
# Logo file name, must be an absolute path without environment
# variables.

8.3.3 Test Execution: Monitoring, Archiving and AP Execution

8.3.3.1 General

During online execution of test, the software running on test nodes (TES) provides automatic data processing and monitoring according to definitions loaded from the MDB. Automated Procedures may be executed and SAS may be controlled that run in parallel to TES on the same node or on remote nodes.
In the following the functions provided by test nodes are described w.r.t. to the interface to the user and to understand the system behaviour.

The test execution activities can be performed in three different modes of operation:

- Normal mode
- Simulation Mode
- Replay Mode

For description of the modes refer to chapter 8.1.

All modes support the monitoring and data processing function, the execution of UCL commands and the visualisation of data. Sending of GDU to SAS is only performed in NORMAL mode (while in other modes the corresponding operations are accepted, but simulated).

8.3.3.2 Monitoring and Data Processing

After having the test node initialised (i.e. having setup the test configuration and the test nodes switched to an operational mode), TES is ready to **acquire** data from SAS. The **SAS** must be setup into operational state using UCL commands. Then the data may be acquired from SAS. Each single item (resp. the whole name tree, a subtree or a monitoring list) may be activated by the START_ACQUISITION command.

TES will then **request** the data from the SAS, supplying the header information and the physical address given in the ADU_DESCRIPTIONs in the MDB. This includes the ADU_ID, the CCSDS Primary Header and the CCSDS Secondary Header information. In case the data is received from onboard links, the SAS will either request the data from this link, using the given information as address information, or just route the data to TES. In case the data is received from frontend devices , most probably the physical address information will be used. ADUs are then sent cyclicly or asynchronously to TES, where the ADUs are processed to fetch the values from the data part.

Data may be received after **interruption**. In the next ADU after the interruption the SAS may signal a problem to TES, which sets the interrupted status then to all enditems referring to this ADU.

The SAS may further indicate an outage problem on ADUs to CGS (via a specific procedure in the TES_API). TES then sets the acquisition status of all enditems of this ADU to ”**static**”.

Enditems may be enabled for **processing** or not. Processing means fetching of raw value, calibration, monitoring and delivering data for display. Processing may be controlled via HLCL/UCL commands or automatically via conditions.

TES will **calibrate** the data as defined for each enditem in the MDB. Calibration means conversion from raw values into engineering values applying a calibration curve. Curves are specified either as polynoms or as point pair sets. For Point Pair Sets , calibration is done by linear interpolation between the point pairs.
Raw values are fetched from the data part of ADUs (of type UNSTRUCTURED or CCSDS) or from the raw value list transferred in Structured ADUs. Raw values are interpreted acc. to the raw value types and bit_size information specified for the measurements.

For CCSDS packet, the system can be configured to verify the checksum of the incoming CCSDS telemetry by setting to true the configuration parameter TES.CHECK_CHECKSUM in the TES_CONFIG_FILE prior starting the system. In that case, if the checksum is found to be incorrect (for those CCSDS packets having a checksum), a message will be generated. Another configuration parameter allows prevent or not the extraction of data from CCSDS packets having an incorrect checksum. This parameter is named TES.PROCESS_ON_INCORRECT_CHECKSUM in the TES_CONFIG_FILE. Of course this configuration parameter only applies in the case where the system is configured to check the checksum.

TES will **archive** the raw data packets (ADU) into archive files, if archiving is enabled. Archive Files are transfered to the central DB Server node as specified when archiving was enabled (default: every 30 minutes)

When received, the ADUs are queued to ensure their processing in the correct order (of reception) and to be tolerant to burst. If the queue ever become full, then the test node was not in the position to handle such a burst and ADU will be discarded (and not processed, but still archived if archiving is enabled). The length of the queue is set by default to 50 ADUs. It can be modified by changing the configuration parameter ADU_QUEUE.MAX_NUMBER_OF_QUEUED_ADUS prior starting the system.

Further enditem values may be received from SAS or APs which write to **software variables**. TES itself may calculate its own values from HK values or generate derived values (see below).

TES will **dispatch** the engineering data to workstations which show them in synoptics. Only those values are transferred which are requested by the loaded synoptics. Together with the engineering value, the acquisition status, processing status and monitoring status is transferred

Monitoring may be activated via the ENABLE_MONITORING command. TES then will monitor each new value according to the active limit sets defined. Exceptions will be generated, if limits are violated.
If Danger Limits are violated, a message of type ”ALRM” is generated. If soft limits (monitor limits) are violated, a message of type MSG is generated for each single violation. If actions are defined for exceptions, they will be executed: APs will be started, GDUs or GDU lists will be sent or user defined messages will be sent to the message handler.

The AP to start from a monitoring exception must not have any parameter, otherwise it will be rejected. The GDU to generate must as well be either without parameter or have default parameters (that are then used). This is due to the fact that the action is executed automatically and that the user has not the possibility to define the parameters.

For the GDUs and GDU list, as in case they are generated interactively, a timeout can be given, a default timeout is used in case they are generated as action from a monitoring exception. This default timeout value can be modified in the TES_CONFIG_FILE under the name GDU_HANDLER.ISSUE_TIMEOUT_WHEN_MONITORING_EXCEPTION prior starting the system.

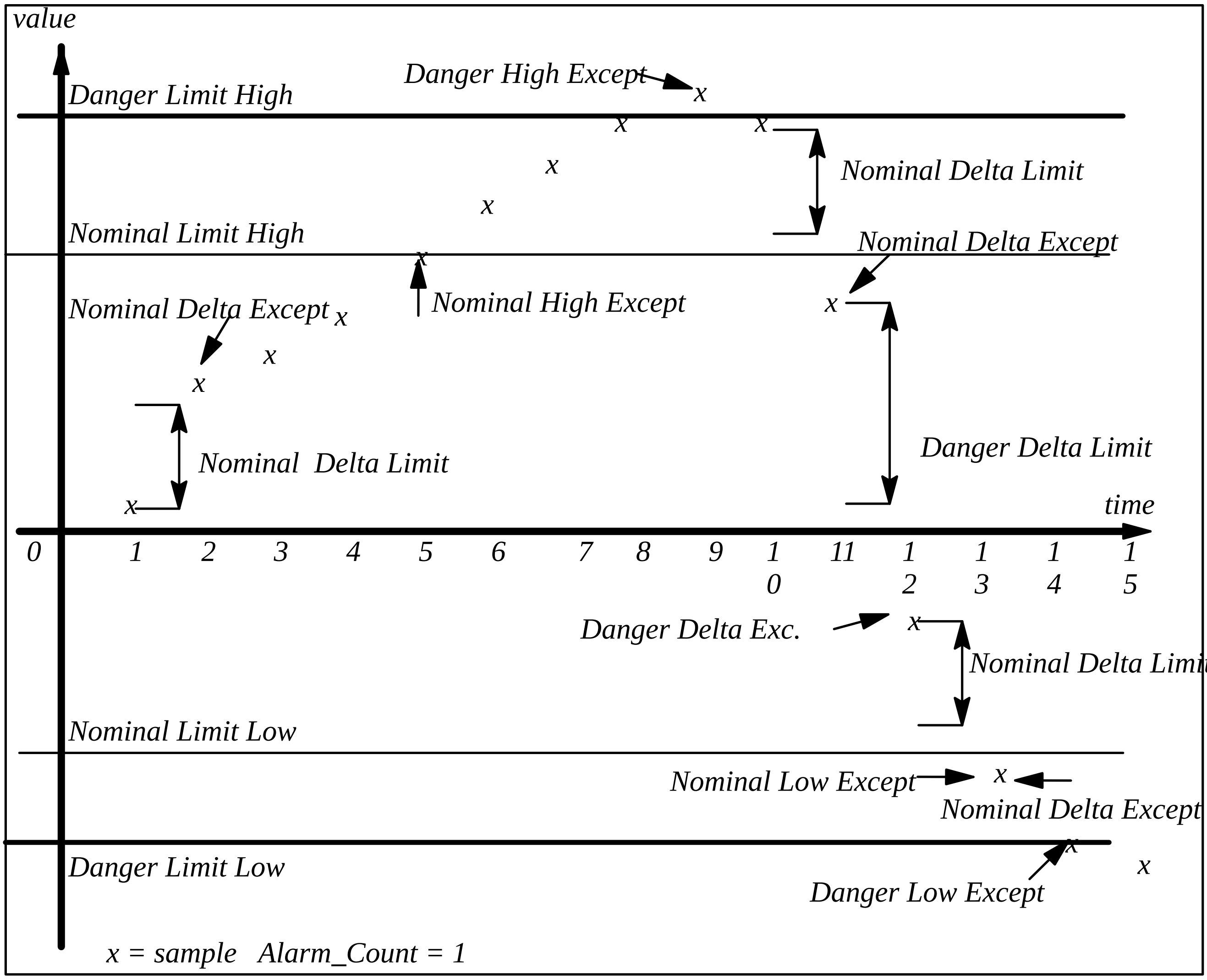
Each monitor limit will have a message allocated which is to be generated by the monitor in case of limit violation (Exception). An exception message is to be handled always as a message to the HCI and to the DBS logging service . A standard exception message is to be generated by TES. This message contains the actual value of the enditem, the limit violated, the applied limit set and a user specified message.
The message is sent as a coded message to HCI.

It is possible to define delta limits and a counter indicating allowed violations of the soft limit set before an exception is generated. A hard (danger) limit violation is always reported.

Range limit checking and delta limit checking are performed in parallel, i.e when monitoring a measurement, the result can be one of the following states:

- nominal

- out of danger limit
- out of danger delta
- out of soft limit
- out of soft delta
- out of danger limit and out of danger delta
- out of danger limit and out of soft delta
- out of soft limit and out of danger delta
- out of soft limit and out of soft delta



The **Alarm Count** specified in the MDB for the enditem is used only for soft(nominal) limit checking. It specifies the number of consecutive out of limit situations before an exception is generated.

General Rules for message/action generation of **analog** values:

Whenever a delta limit is violated (and was not violated in the previous sample and no interruption of data acquisition occurred), a message/action is generated. When a danger delta limit is violated, no nominal delta message/action is generated.

Whenever the value crosses the limit border (from in_limit to out_of_limit) between two samples, a corresponding message/action is generated. If both the nominal and danger limit borders are crossed, only one message/action shall be generated (for danger limits). For nominal limits, the msg/action shall be generated after ALARM_COUNTER samples are still in that same out of limit state.

After a danger message/action is generated, the next danger message/action can only be generated in one of the following cases:

- The measurement goes from above danger high limit to below danger low limit.
- The measurement goes from below danger low limit to above danger high limit.
- The measurement leaves for at least one sample the out of danger limit area (into out of nominal limit or into nominal range) before going back into the out of danger limit area.

After the generation of a message/action for a danger high limit violation, the generation of message/action from out of nominal high limit is inhibited until the measurement goes at least for one sample below the nominal high limit.

After the generation of a message/action for a danger low limit violation, the generation of message/action from out of nominal low limit is inhibited until the measurement goes at least for one sample above the nominal low limit.

After the generation of a message/action for a nominal high limit violation, the generation of a new message/action from out of nominal high limit is inhibited until the measurement goes at least for one sample below the nominal high limit.

After the generation of a message/action for a nominal low limit violation, the generation of message/action from out of nominal low limit is inhibited until the measurement goes at least for one sample above the nominal low limit.

After a danger delta message/action is generated, the next danger delta message/action is generated only if in between the delta monitoring of the measurement indicates either a nominal status or an out of soft delta status.

After a danger delta message/action is generated, the generation of a nominal delta message/action is inhibited until the delta monitoring of the measurement indicates a nominal status.

After a nominal delta message/action is generated, the generation of a nominal delta message/action is inhibited until the delta monitoring of the measurement indicates a nominal status.

Rule for **discrete** values:

Whenever the value is not as the expected value, and was as the expected value in the sample before ALARM_COUNTER samples, but was different for all the previous ALARM_COUNTER samples, a message/action is generated.

In case the **interruption** flag is set for an ADU, or the ADU has been indicated as **static**, the resp. enditems shall not be subject to delta monitoring until two consecutive items without interruption flags are received.

<u>SW Variables</u>

SW Variables are enditems that are generated within software. The source of the value is an AP or SAS writing to it or an internal HK value that is written by TES. The values are updated whenever a request is received from AP or SAS. SW Variables can be monitored in the same way as measurements.

HK values

The TES will provide the update service for each of its housekeeping variables.

HK values will be referenced to the user (HCI, AP or SAS) via SW variable: each HK value has an internal identifier. This identifier is given in the definition of a SW variable. TES will then provide the HK value whenever a user refers to the SW variable. Through this, it is also possible to monitor the value of housekeeping variables (e.g. free disk space) and to take appropriate actions in case the value goes out of limit.

Some HK values are modified on change (the one that are directly under the control of the test node software). Other HK values are modified cylcicly. The update cycles of those HK values is defined in the TES_CONFIG_FILE and their values can be modified inside that file prior starting the system. The configurable parameters of that file are HK_VALUE_PROVIDER.UPDATE_CLOCK_PERIOD for the HK values related to the time, HK_VALUE_PROVIDER.UPDATE_DISK_PERIOD for the HK values related to the local disk space, HK_VALUE_PROVIDER.UPDATE_DBS_PERIOD for the HK values related to the status of the test result database and the database server, HK_VALUE_PROVIDER.UPDATE_TSS_PERIOD for the HK values related to the SMT and its status and UCLI_CONTROLLER.HK_VALUE_UPDATE_PERIOD for the UCL related HK values.

Note, especially for the last parameter that a high frequency may lead to additional CPU load of the test node.

Derived Values

Derived Values can be defined in the MDB that define enditem values as a result of an expression on other enditem values. Standard UCL is used to define the expression in the Database. Expressions are converted to I–Code and executed as any other UCL code. Some restrictions apply to the expressions (see ch. 7.6). The expression is calculated whenever a new value is received (via ADU or as a result of write operation to software variables).

Derived Values may refer to other Derived Values.

A derived value that is referencing several measurement from one ADU will only be calculated once when the ADU is processed because derived values referencing measurements are calculated at the end of the processing of the ADU once all measurements have been processed.

Enditem Grouping

Enditems (measurements, sw variables and derived values) will be addressable as a set (group). This will be implemented by taking the name hierarchy in the DB and the monitoring lists:

- o An enable/disable command with incomplete name–tree information (i.e. specifying a name that is not on the leaf level) will be interpreted in such a way that all parameter below that name are to be enabled/disabled.
- o An enable/disable command giving a monitoring list (in UCL: ’monitoring table’) as a command parameter will be interpreted to enable/disable all enditems given in the list.
- o Both methods may be used in parallel, thus allowing specification of all enditems under a specified subtree which belong to a specified monitoring list.

Grouping can be applied to many operations on measurements, sw variables and derived values.

Conditions

Conditions may be specified that allow to enable/disable processing of enditems, defining the applied limit sets of enditems or start APs. Conditions are true or false when a specified expression becomes true or false. An expression is a simple expression of an enditem with a comparator and a value (e.g. ”\enditem\a >= 0”).

Conditions are defined for an enditem whose value triggers the action. Several Conditions may be active in parallel for the same enditem. Actions may refer to single enditems or groups (virtual trees, monitor lists).

Whenever a condition is triggered, a message is produced in the Event Logging, not in the message window. In case the system is heavily using conditions and the messages are not desired, their generation can be disabled by modifying the parameter DATA_PROCESSOR.MESSAGE_ON_CONDITION in the TES_CONFIG_FILE prior starting the system.

<u>Online Modification</u>

It is possible to enable/disable the limit checking for a parameter by an online HLCL/ UCL command. Together with the enable/disable command, the limit set to be applied may be given that overwrites the set given as ’default set’ in the measurement definition. New limit values may be specified online via UCL. The modifications are valid until the test node is reinitialised with forced reloading (see Test Setup)

8.3.3.3 Sending of Generation_Data_Units (GDU)

TES loads the following enditem types from the MDB (LOAD_SCOE):

EGSE_ANALOG_STIMULUS
EGSE_DICRETE_STIMULUS
EGSE_BINARY_PACKET
EGSE_PREDEFINED_TC
SWOP_COMMAND (specific for the COF project)

All these enditems are translated to GDUs before sending them to the SAS specified.
Stimuli allow to send single values, while the remaining allow to send a packet of values: either in a binary buffer or as a CCSDS packet. (EGSE_PREDEFINED_TC, SWOP_COMMAND).

TES generates the actual **packet** from the information given in the MDB (CCSDS Header, CCSDS Secondary Header (COF), Predefined Data. The packet is completed by parameter given online in the UCL ISSUE command. The parameter values are inserted into the packet at specified places, overriding possibly the predefined values.

A GDU is **sent to the SAS** and then acknowledged by the SAS. If the acknowledge is not received within a configurable timeout value, an error message is generated and the failure is indicated to the caller of the ISSUE statement.

For SWOP_COMMANDs, a specific return package (response packet) may be specified. For description refer to the CGS ICD and the Ground_to_Onboard Library given in Appendix I of this document.
SWOP–COMMANDs are converted to GDUs, and response packets are handled as normal ADUs. Thus they are archived as any other ADU/GDU. The configurable parameter SW_CMDER.ISSUE_TIMEOUT_WHEN_NO_DELAY in the TES_CONFIG_FILE allows to specify a GDU timeout for software commands in case the timeout from the UCL command is set to zero (meaning no response packet is expected).

Any sending of a GDU is logged as an event. Any GDU is archived, if archiving is enabled

<u>Critical commands</u>

It is possible to save telecommands (GDU’s except SWOP_COMMAND) against unauthorized sending. Each single telecommand can be defined in the mission database together with a flag “Critcal command” and an optional password. This password is a string of up to eigth printable ASCII–characters.

If the telecommand is marked as “Critcal command”, each sending must be authorized by confirm (no password defined) or authorized by password. The user is requested in HCI to allow the sending of

telecommand. Three tries are allowed for each password request. If no HCI is connected to TES, the sending of critical commands is rejected.

All unauthorized tries are logged to error message handler.

Enditem Grouping : GDU lists

It is also possible to define GDU lists that are a list of individual GDU's. When issuing a GDU list, the test node software will issue one GDU after the other from that list.

Priority

GDUs can be issued via a UCL command with a low or a high priority. A third "emergency" priority exists but is not available to the user via the UCL command and is reserved for the GDUs to be issued as a result of a monitoring exception.

For every SAS to which GDUs are sent, three HDU queues are existing (low, high and emergency) and they are treated according to their priority.

CCSDS time

For the CCSDS time that is used in the CCSDS commands, an epoch start time can be specified in the TES_CONFIG_FILE. The three configurable parameters GDU_TABLE.TAI_EPOCH_START_YEAR, GDU_TABLE.TAI_EPOCH_START_MONTH and GDU_TABLE.TAI_EPOCH_START_DAY can be modified inside that file prior starting the system.

CCSDS Sequence counter

Every testnode manages the CCSDS sequence counter to be set for CCSDS telecommand, depending on the application id of the primary header. The configuration parameter in the TES_CONFIG_FILE REQUEST_CONNECTOR.USE_TYPE_IN_CCSDS_SEQ_COUNT is used to indicate if the packet type field of the CCSDS primary header has to be taken into account for the application ids. By default that field is taken into account.

Online Modification

It is possible to modify the reference of the SAS to which the GDU is to be sent using a UCL command ("routing"). The modifications are valid until the test node is reinitialised with forced reloading (see Test Setup)

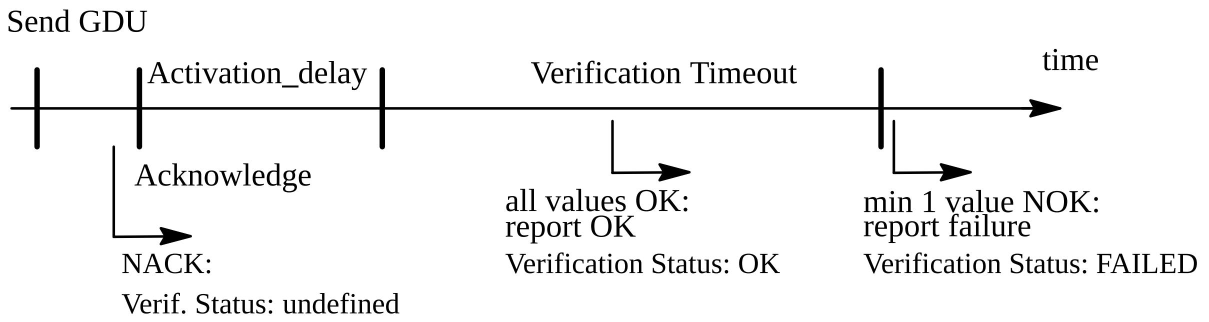
8.3.3.4 Telecommand verification

It is possible to defined in the MDB automatic verifications to apply after having sent GDU of the following types :

EGSE_ANALOG_STIMULUS
EGSE_DICRETE_STIMULUS
EGSE_BINARY_PACKET
EGSE_PREDEFINED_TC

This definition includes two times, the activation delay and the verification timeout as well as a set of measurements, software variables or derived values to be verified against a given value or range.

After the GDU is successfully issued, the test node will start the telecommand verification after the specified activation delay. It will then check the value of the measurements, software variables or derived values against the specified value or range. In case all checks are successful, the TC verification is then successful. In case the verification is not successful, every time the value of the measurements, software variables or derived values is modified, the checks are made. If after the specified verification timeout, the verification is still not successful, it has then failed.



When issuing a GDU, the TC verification will be asynchronously triggered if defined, ie the UCL Ground library routine used to send the GDU will not wait until the verification is done.

A UCL Ground library routine allows to check the verification status of a GDU.

A UCL Ground library routine allows to issue a GDU overwriting the specified times (activation delay and verification timeout). This operation can as well be used in a synchronous mode where it completes only at the end of the TC verification.

In TES_CONFIG_FILE the TC verification of all telecommands can be disabled by setting parameter:
GDU_HANDLER.TC_VERIFICATION_DISABLED 4 true.

8.3.3.5 Archiving and Logging

Archiving
TES archives the following items into archive files:

- ADUs (Unstructured, Structured, CCSDS Packets)
- GDUs (Stimuli, Telecommands, Binary Packets)
- Changes in the SMT
- Requests for ADUs

Archiving may be enabled/disabled for the whole test node. Archiving cycles (i.e. periods after which the archive file is closed and transferred to the TRDB) may be specified when enabling the archive.
Archive files may be closed on request (CLOSE_ARCHIVE).

Archive Files can be evaluated later or during the test session using the TEV tool, generating a dump of the ADUs/GDUs in hexadecimal or ASCII format, or by recalibrating measurement values from the ADUs.

Archiving files are the basis for the replay of test sessions in the REPLAY mode.
Archive Files when closed are transferred to the TRDB and handled as part of the test session.

Event Logging
Any user related action in TES is logged as an event in the Event Log of the TRDB. User Messages are sent in addition to the HCIs connected to the TES, and thus are logged in the log files of the message handler.

Event Logs can be evaluated by the TEV tool after or during the test session to generate complete or selected event lists. Message Logfiles can be viewed by the message handler during or after the ongoing test.

Engineering Value Logging
TES will **log** each new engineering value if defined so for this enditem and if the value is different from the

previous one. Enabling/disabling of engineering value logging (EVL) is possible via UCL commands, either on individual enditems or on virtual trees / monitoring lists. EVL can be used later during evaluation to generate datasets and then data listings, graphical curves or Excel spreadsheets.

8.3.3.6 UCL Execution

UCL Language

UCL is defined as a set of Test object or Test system oriented commands together with a set of control commands allowing specification of automatic procedures with conditional execution. (UCL is used for both Onboard and Ground purposes).

The UCL Language

- Provides type conversions
- Supports engineering units
- Supports CGS pathname concept
- Conditional statements
  – IF, CASE, LOOP, WHILE, etc
- Libraries
  – user libraries and system libraries

UCL allows for definition of Automated Procedures (APs) that may be activated by an interactive command and can call other APs. Each AP is compiled offline during test preparation by the UCL Compiler. This generates an intermediate language format called i–Code. This is interpreted by the UCL Interpreter called by the Test Execution Software.

UCL allows to define User Libraries implementing additional procedures that can be called from any AP or HLCL command window. User Libraries are loaded to a test node. and executed as I–Code.

UCL system libraries are specification of services/procedures implemented in TES. System library specification must be imported into APs or User Libraries to allow them to be used.

The UCL GROUND_LIBRARY System Library provides a broad range of function for following topics:

- **Data Processing** control
  – Start/Stop acquisition of enditems
  – define conditions to control processing
  – get status on conditions
  – get acquisition status
- **Monitoring** control
  – change online limits
  – enable/disable
  – get monitor status
- **Time** management
  – get local time and simulated mission time (SMT)
  – SMT management
  – wait
- **Automated Procedure** control
  – start another automated procedure (asynchronous, synchronous)

  – suspend/resume automated procedure
  – get status of automated procedure
  – exchange messages between automated procedures

- **SAS** control
  – start / stop a SAS
  – change SAS current mode (init, reset, etc.)
  – get status of SAS
  – exchange messages with SAS (synchronous, asynchronous)
- **Stimulus(GDU)** generation
  – send stimulus/TC
  – send a list of stimuli/TC
  – enable/disable certain stimuli/TC
- Log & **Event** handling
  – generate a log event
  – user events
  – enable/disable engineering value logging
- **Archive** control
  – enable / disable archiving
  – define archiving cycle
  – close the archive
- **Synoptic** display control
  – show a dedicated synoptic on a dedicated screen
  – remove a synoptic from the screen
- **User input & output**
  – write a message to the user
  – read a message from the user
- Control of **Conditions**
  – define/delete conditions online
  – enable/disable conditions
  – interrogate condition status
- General **conversion** procedures and functions

Refer to Appendix I for a specification of the Ground Library.

The writing to **text files** and reading from text files is implemented via a specific user library (**FILE_IO_LIB**). Refer to Appendix M for further description.

Other System Libraries are the **TCL Library** and the **Ground_Commands_to_Onboard** library, which serve specific purposes for specific projects.

<u>AP Execution</u>

Up to 40 APs may be executed in parallel on each test node. Each AP is executed within one slot of the AP Executor. By default only 20 APs can be executed in parallel. The number can be decreased down to 1 / increased up to 40 by modifying the configuration parameter UCLI_CONTROLLER.NUMBER_OF_UCL_INTERPRETER in the TES_CONFIG_FILE prior starting the system. It must be noted that increasing the number of APs that can be executed in parallel leads to an increase of memory

required by the test node software as for every slot, memory must be allocated for loading and executing the AP.

By default, 5 slots are reserved for so called emergency APS. i.e. APs that are started by the monitoring/condition function. In case a emergency AP is running, all other non–emergency APs are suspended. This number of slots reserved for monitoring exception can be modified in the TES_CONFIG_FILE, parameter UCLI_CONTROLLER.MAXIMUM_NUMBER_OF_EMERGENCY_APS prior starting the system.

All APs can be aborted by one single HLCL command ABORT_ALL_APS.

APs can be suspended and resumed for execution. The actual status of the AP can be monitored in the AP Status window.

If defined so in the TES Configuration File, TES keeps the actual statement of an executing AP in a specific HK value, which can be monitored in the AP status window. Furthermore, the executed I–Code can be logged in a specific file, as defined in the TES Config. File. This allows for minimum AP debugging support during AP development.

A CGS User involved in Testing operations will be able to issue **interactive commands**, from the Workstation, to distributed test software via appropriate User interaction methods such as windows, menus, dialog boxes etc. These commands encompass UCL statements and other Workstation or Test specific commands which together form the High Level Command Language (HLCL).

The same AP can be started several times in parallel, except for the AP's started as result of a monitoring exception or as result of a condition, that will only be started if not already running.

<u>Priority</u>

AP's can be started via a UCL command with a low or a high priority. A third "emergency" priority exists but is not available to the user via the UCL command and is reserved for the APs to be issued as a result of a monitoring exception.

The scheduling of the AP's on one test node is an optimised preemptive scheduling model. All AP's of the same priority level are executed in parallel. Whenever an AP of a given priority level is running, AP's of lower priority levels are suspended. They are automatically resumed when no AP of a higher priority level is running any more.

<u>Online Modification</u>

UCL APs and user libraries may be reloaded from the MDB after re–compilation without re–initialising the test node. This is done using a Ground library routine. Thus modification of APs during online testing is supported. The modifications are valid until the test node is reinitialised with forced reloading (see Test Setup).

8.3.3.7 Communication with SAS

Test Nodes may communicate with a set of SAS, running on the same node or on remote nodes.
SAS may receive GDUs or may deliver ADUs. SAS also may write to software variables or read raw or engineering values from the testnode.

SAS can be controlled interactively or from APs/HLCL Sequences, using the command implemented in the UCL Ground Library.

The following limitations exist:

- Up to 20 SAS can be active for the same test node.

Different configuration parameters in the TES_CONFIG_FILE can be used to tailor the test node to the system needs. Timeout values can be specified for the different commands sent from the test node to the SASes. Those configuration parameters are : TES.ENABLE_ACQUISITION_TIMEOUT, TES.DISABLE_ACQUISITION_TIMEOUT, TES.LOAD_APPLICATION_TIMEOUT, TES.INIT_APPLICATION_TIMEOUT, TES.START_APPLICATION_TIMEOUT, TES.RESET_APPLICATION_TIMEOUT, TES.GET_APPLICATION_STATUS_TIMEOUT, TES.WRITE_MESSAGE_TO_APPLICATION_TIMEOUT, TES.DOWNLOAD_FILE_TO_APPLICATION_TIMEOUT and API_CONTROLLER.TIMEOUT_PERIOD_FOR_READ_CMD.

8.3.3.8 Replaying data

In Replay mode, the data is replayed from the archive files produced during a test and specified during the replay setup:

- ADUs (Unstructured, Structured, CCSDS Packets)
- GDUs (Stimuli, Telecommands, Binary Packets)
- Changes in the SMT
- Requests for ADUs

The data from the event log and engineering value log are not replayed. Neither the AP’s and sequences that have been started during the test are replayed.

When an ADU request is detected that was a start acquisition command, all measurements from that ADU will be processed from that ADU.

When an ADU request is detected that was a stop acquisition command, the acquisition of all measurements from that ADU will be stopped.

When an ADU is replayed from an archive file, the measurements it contains that are acquired will be processed, as during the normal mode (see Monitoring and Data Processing). The measurements can be acquired by replaying a start acquisition command or using the UCL commands as during the normal mode. Note that as it can occur that the start acquisition commands are not replayed (e.g. due to a restriction of the time frame to replay), a configuration parameter in the TES_CONFIG_FILE allows to automatically enable all measurements in the test nodes to avoid having to enable everything via the UCL command. That configuration parameter is called DATA_PROCESSOR.ENABLE_ALL_ADUS_IN_REPLAY and must then be set to true prior starting the system.

When a GDU is replayed from an archive file, a message will be issued to the user. In case this would lead to a too large number of messages and this is not wanted, the generation of the messages can be prevented by setting the configuration parameter REPLAYER.GDU_MESSAGES to false in the TES_CONFIG_FILE prior starting the system.

When a change of the SMT (start or stop SMT command) is replayed from an archive file on the MTP, the SMT will be accordingly set during the replay session.

The test node is managing the replay time according to the replay time frame and the speed factor specified during the setup phase. As the SMT is a system wide time, in case the speed factor is different from 1, in opposition to the internally managed replay time, the SMT value will ”run” at a speed factor of 1. The MTP will then cyclically ”re–synchronise” the SMT to the replayed value in case of deviation. This synchronisation will be performed by default every 5 seconds. This value can be modified in the configuration parameter SMT_UPDATE_PERIOD_IN_REPLAY of the TES_CONFIG_FILE prior starting the system.

During replay mode, all other functionality in the test node are available as during normal mode, i.e. AP's can be executed, monitoring can be done, etc.

A few differences are in the fact that some UCL library routines will be ignored, as for example the starting or stopping of SMT, the generation of GDU, the enabling or disabling of archiving.

8.3.3.9 Simulating data

In simulation mode, the test node has the same functionality as in normal mode except that the ADU's are not acquired from a SAS and the GDU's are not sent to a SAS.

The sending of a GDU will result in the generation of a message.

In simulation mode, the ADU's are simulated. ADU that will be required to be simulated must have in the MDB a simulated ADU description. UCL commands allow in simulation mode to set specific raw value (enditem values) of a structured ADU or to set specific data in an unstructured ADU or in a CCSDS packet.

8.3.3.10 Internode Communication

On test nodes, data may be acquired from other nodes by simply referencing enditems loaded to those nodes. TES manages to route the data to the actual node.

The following is supported:

- Starting of APs on remote nodes
- Reading of enditem values (measurements, software variables or derived values) from remote nodes
- Writing Software variable from remote nodes
- Sending GDU (Stimuli) or GDU lists via remote nodes
- reading the TC verification status from a remote GDU

Starting an AP remotely is performed by specifying on which node the AP has to be started while the remaining remote operations are transparent (any test node knows which test node is maintaining a given measurement, software variable, derived value or GDU).

The following limitations exists:

- Monitoring attributes cannot be interrogated nor set on remote nodes
- Acquisition, Monitoring, Logging and Archiving can be enabled only on local nodes
- APs can communicate only to SAS having been started and assigned to the local node.

In order to be able to have internode communication, each testnode must maintain references of the data maintained by remote nodes (references to measurements, software variables, derived values, GDUs and GDU lists). The test nodes synchronise themselves automatically during the test setup. The references are held in a table in each test node. The length of that table has been defined to be able to have up to 20000 external references by default. This value can be adapted to special needs in case a system is maintaining more data by modifying the parameter DISTRIBUTION_TABLE.MAX_NB_SID in the TES_CONFIG_FILE prior starting the system.

8.3.3.11 The TES_CONFIG_FILE

TES behaviour can be controlled by various configuration parameters defined in the file TES_CONFIG_FILE located under the directory $GSAF_HOME/tes/config. The file has many parameters. In the normal case there is no need for changing the parameters, because appropriate default values has been defined, except for 2 parameters (SW_CMDER.EXEC_FLAP_SWOP_PATHNAME and SW_CMDER.EXEC-WAIT_FLAP_SWOP_PATHNAME) that are specific for the COF project. However an 'advanced' user may want to change some of the parameters to adapt the TES behaviour to his needs. Many of the parameters has only interest for the engineers and the developers of the system (marked with 'E'). Some of them, marked with 'U', the advanced user should know about. In the following the most important of them are described, also those marked with 'E'.

The remaining parameters, not described here must not be modified.

Timeouts for SAS commanding:

```
!! (U) Timeout for ack from SAS upon enable ADU request, seconds
!!    Range: float'range
!!    Recommended value: 60.0
TES.ENABLE_ACQUISITION_TIMEOUT        2    60.0
!!
!! (U) Timeout for ack from SAS upon disable ADU request, seconds
!!    Range: float'range
!!    Recommended value: 60.0
TES.DISABLE_ACQUISITION_TIMEOUT       2    60.0
!!
!! (U) Timeout for connect from SAS after being started, seconds
!!    Range: float'range
!!    Recommended value: 30.0
TES.LOAD_APPLICATION_TIMEOUT          2    30.0
!!
!! (U) Timeout for ack from SAS upon init application, milliseconds
!!    Range: positive'range
!!    Recommended value: 60_000
TES.INIT_APPLICATION_TIMEOUT          1    60_000
!!
!! (U) Timeout for ack from SAS upon start application, milliseconds
!!    Range: positive'range
!!    Recommended value: 60_000
TES.START_APPLICATION_TIMEOUT         1    60_000
!!
!! (U) Timeout for ack from SAS upon reset application, milliseconds
!!    Range: float'range
!!    Recommended value: 60_000
TES.RESET_APPLICATION_TIMEOUT         1    60_000
!!
!! (U) Timeout for ack from SAS upon get application status, milliseconds
!!    Range: float'range
!!    Recommended value: 60_000
```

```
TES.GET_APPLICATION_STATUS_TIMEOUT          1    60_000
!!
!! (U) Timeout for ack from SAS upon write message to application, milliseconds
!!     Range: float'range
!!     Recommended value: 60_000
TES.WRITE_MESSAGE_TO_APPLICATION_TIMEOUT     1    60_000
!!
!! (U) Timeout for ack from SAS upon download file to application, milliseconds
!!     Range: float'range
!!     Recommended value: 60_000
TES.DOWNLOAD_FILE_TO_APPLICATION_TIMEOUT     1    60_000
!!

!!
!! (U) If an SAS does not call read_command within this time period, then
!! TES will automatically disconnect the SAS
!!     Range: duration'range
!!     Recommended value: 30.0 s
API_CONTROLLER.TIMEOUT_PERIOD_FOR_READ_CMD   2    30.0
!!
```

Parameters for test node setup:

```
!!
!! (U) Default workstation (HCI) for TES output in emergency case.
!!     Must be a logical name (see SYSTEM_TOPOLOGY_TABLE).
!!     Range: array (1..255) of character
!!     Recommended value: HCI_01
TES.DEFAULT_WORKSTATION                 3     "HCI_01"
!!

!!
!! (E) Time-out for the INIT operation of the TES_RPI. It may be necessary
!!     to increase that value when loading very large databases. The time-out
!!     is expressed in milliseconds.
!!     Range: integer'range
!!     Recommended value: 300_000
TES_RPI.TIME_OUT_INIT                   1    300_000
!!

!!
!! (E) Maximal number of enditems in the test configuration.
!!     This number constrains a list that is used for remote
!!     operations on measurements / sw-variables, GDU's and GDU lists.
!!     Range: integer'range
!!     Recommended value: 20_000
DISTRIBUTION_TABLE.MAX_NB_SID                1     20_000
!!
```

for the following setup parameters, modifications shall only be done in coordination with CGS development team:

```
!!
!! (U) Timeout for ack from another TES in initialisation
!!     exchange of distribution tables), milliseconds
!!     Range: integer'range
!!     Recommended value: 30_000
TES.INITIALISE_TIMEOUT     1    30_000
!!

!!
!! (E) Stack size of the init_data_server task
!!     Range: integer'range
!!     Recommended value: 15_000_000
!!INIT_DATA_SERVER.STACK_SIZE                 1 15_000_000
!!
```

Parameters for data processing:

```
!!
!! (U) The maximum number of queued ADUs before TES starts to discard
!!     Range: integer'range
!!     Recommended value: 50
ADU_QUEUE.MAX_NUMBER_OF_QUEUED_ADUS         1    50
!!

!!
!! (U) Indicates if the time stamp of the measurements (accessible via UCL)
!!     is based on local time (true) or SMT (false).
!!     Range: boolean'range
!!     Recommended value: true
DATA_PROCESSOR.MEASUREMENT_TIME_STAMP_IN_LT          4       true
!!

!!
!! (E) allow or inhibit the generation of messages to the event log
!!     for each triggered condition.
!!     Range: true/false
!!     Recommended value: true
DATA_PROCESSOR.MESSAGE_ON_CONDITION                  4    true
!!
!! (E) allow or inhibit the checking of the checksum in ADU of type CCSDS
!!     packet or in response packet for software commanding. If it is set to
!!     true and the checksum is not detected to be
!!     correct, an error message is generated.
!!     Range: true/false
!!     Recommended value: false
TES.CHECK_CHECKSUM                                   4    false
!!
!! (E) allow or inhibit the processing of measurements contained in ADU of type
!!     CCSDS packets or the processing of response packets for software
!!     commanding if the checksum is incorrect.
```

```
!!     This parameter is only used in the case where TES.CHECK_CHECKSUM is
!!     set to true.
!!     Range: true/false
!!     Recommended value: true
TES.PROCESS_ON_INCORRECT_CHECKSUM                    4    true
!!
```

Monitoring action Parameters: modifications shall only be done in coordination with CGS development team

```
!!
!! (E) Number of action_handlers in the monitor, i.e. the number of monitor
!!     actions (GDUs or APs) that can be handled in parallel
!!     Range: integer'range
!!     Recommended value: 5
MONITOR.ACTION_HANDLERS                  1         5
!!

!!
!! (E) The length of the action_buffer data structure
!!     Range: integer'range
!!     Recommended value: 100
ACTION_BUFFER.ACTION_BUFFER_BUFFER_LENGTH    1    100
!!
```

Update periode for housekeeping values:

```
!! (U) Time period for cyclical update of the HK value for LT
!!     In replay it will be used to set the SMT in the HK data
!!     Range: duration'range
!!     Recommended value: 1.0
HK_VALUE_PROVIDER.UPDATE_CLOCK_PERIOD       2      1.0
!!
!! (U) Time period for cyclical update of the HK value for free disk space
!!     Range: duration'range
!!     Recommended value: 30.0
HK_VALUE_PROVIDER.UPDATE_DISK_PERIOD        2      30.0
!!
!! (U) Time period for cyclical update of the HK values from DBS
!!     Recommended value: 60.0
HK_VALUE_PROVIDER.UPDATE_DBS_PERIOD         2      60.0
!!
!! (U) Time period for cyclical update of the HK values from TSS
!!     Recommended value: 60.0
HK_VALUE_PROVIDER.UPDATE_TSS_PERIOD         2      60.0
!!

!!
!! (E) Period for writing the "current UCL statement" HK value, seconds
```

!! Range: duration'range
!! Recommended value: 5.0
UCLI_CONTROLLER.HK_VALUE_UPDATE_PERIOD 2 5.0
!!

CCSDS–Recommended epoch start time:

!!
!! (U) The CCSDS–Recommended epoch start time (TAI)
!! Range: integer'range
!! Recommended value: 1_980
GDU_TABLE.TAI_EPOCH_START_YEAR 1 1_980
!!
!! (U) The CCSDS–Recommended epoch start time (TAI)
!! Range: integer'range
!! Recommended value: 1
GDU_TABLE.TAI_EPOCH_START_MONTH 1 1
!!
!! (U) The CCSDS–Recommended epoch start time (TAI)
!! Range: integer'range
!! Recommended value: 6
GDU_TABLE.TAI_EPOCH_START_DAY 1 6
!!

GDU Parameters:

!!
!! (U) Default timeout value when issuing gdus upon a monitoring
!! exception, seconds
!! Range: duration'range
!! Recommended value: 5.0
GDU_HANDLER.ISSUE_TIMEOUT_WHEN_MONITORING_EXCEPTION 2 5.0
!!

!!
!! (U) indicates if the type field should be considered for getting
!! sequence count or not
!! Range: boolean'range
!! Recommended value: true
REQUEST_CONNECTOR.USE_TYPE_IN_CCSDS_SEQ_COUNT 4 true
!!

!!
!! (U) Default value for disable tc_verification to forbid
!! the verification of all sended TC's independ of MDB definitions
!! Range: boolean'range
!! Recommended value: false
GDU_HANDLER.TC_VERIFICATION_DISABLED 4 false
!!

Software Commanding Parameters:

!!
!! (U) Pathname for the SW command to be used for starting a FLAP in
!!   the onboard system without waiting for the FLAP to terminate.
!!   Range: array (1..255) of character
!!   Recommended value: No
SW_CMDER.EXEC_FLAP_SWOP_PATHNAME 3 ”\<PATHNAME>”
!!

This is the name of the software command that is used for starting a FLAP (through the GROUND_COMMANDS_TO_ONBOARD operation EXECUTE_FLAP).
Only when using software commands, it is necessary to have that parameter set in the config file, in that case the correct pathname of the corresponding software command must be set in the file (in upper case). When software commands are not used, the parameter can be left empty (value ””), removed from the config file or be set to any value.

!!
!! (U) Pathname for the SW command to be used for starting a FLAP in
!!   the onboard system with subsequent waiting for the FLAP to terminate
!!   Range: array (1..255) of character
!!   Recommended value: No
SW_CMDER.EXECWAIT_FLAP_SWOP_PATHNAME 3 ”\<PATHNAME>”
!!

This is the name of the software command that is used for starting a FLAP (through the GROUND_COMMANDS_TO_ONBOARD operation EXECWAIT_FLAP).
Only when using software commands, it is necessary to have that parameter set in the config file, in that case the correct pathname of the corresponding software command must be set in the file (in upper case). When software commands are not used, the parameter can be left empty (value ””), removed from the config file or be set to any value.

!!
!! (U) Check that the transaction id of the response packet matches the
!!   CCSDS primary header of the SW command sent
!!   Range: boolean’range
!!   Recommended value: false
SW_CMDER.CHECK_TRANSACTION_ID 4 false
!!

When this parameter is set, the transaction_id returned by the operations ISSUE_SW_COMMAND, EXECUTE_FLAP and EXECWAIT_FLAP is checked and produce an error message in the case where the value differs from the expected value.

!!
!! (U) Default timeout for telecommand part value when issuing a software
!!   command with null delay, seconds
!!   Range: duration’range
!!   Recommended value: 5.0
SW_CMDER.ISSUE_TIMEOUT_WHEN_NO_DELAY 2 5.0
!!

UCL Parameters:

```
!!
!! (E) Enables debug output from I–code interpretation; one file per AP
!!    Range: boolean'range
!!    Recommended value: false
!!STACK_MACHINE.DEBUG                    4    false
!!
!! (E) number of ap interpreters. This determines the number of AP that
!!    can run in parallel.
!!    Range: 1 .. 40
!!    Recommended value: 20
UCLI_CONTROLLER.NUMBER_OF_UCL_INTERPRETER   1        20
!!
!! (E) The number of slots that shall be reserved for emergency APs
!!    Range: 1..UCLI_CONTROLLER.NUMBER_OF_UCL_INTERPRETER
!!    Recommended value: 5
UCLI_CONTROLLER.MAXIMUM_NUMBER_OF_EMERGENCY_APS      1     5
!!
```

for the following UCL parameters, modifications shall only be done in coordination with CGS development team:

```
!!
!! (E) Stack size of the common i–code pool task
!!    The stack is needed when loading i–code from the database.
!!    Using very large i–code may require increase of this stack size
!!    Range: integer'range
!!    Recommended value: 500_000
!!COMMON_POOL.STACK_SIZE                   1   500_000
!!
!! (E) Stack size of the ap_controller.stack_machine_initiator task
!!    Loading very large i–code may require increase of this stack size
!!    Range: integer'range
!!    Recommended value: 40_000
!!AP_CONTROLLER.STACK_MACHINE_INITIATOR.STACK_SIZE       1   40_000
!!
```

Archiving Parameters:

```
!!
!! (U) Limit for issuing warning concerning the amount of free disk space
!!    Range: natural'range
!!    Recommended value: 20_000
ARCHIVE.LIMIT_DISK_SPACE                1   20_000
!!
```

Replay mode Parameters:

```
!!
!! (E) Enables all ADUs in the init–call
```

```
!!    Range: true/false
!!    Recommended value: false
DATA_PROCESSOR.ENABLE_ALL_ADUS_IN_REPLAY          4    false
!!

!!
!! (U) Indicates if the log events / messages produced during a replay session
!!    use the replay local time or the current local time.
!!    Range: boolean'range
!!    Recommended value: false
REPLAY_TIME_FOR_EVENT_TIME_STAMPS          4      false
!!


!!
!! (U) Enables the generation of an error message for each GDU that is replayed
!!    from the archive files. This has no impact on messages issued when
!!    generating a GDU from the replay sesion (e.g. as response of a monitoring
!!    exception or from an AP).
!!    This should be set to false when replaying sessions with a lot of GDU's,
!!    especially when the replay is accelerated.
!!    Range: boolean'range
!!    Recommended value: true
REPLAYER.GDU_MESSAGES                      4      true
!!
!! (E) The frequency to update the SMT clock in replay mode
!!    Range: duration'range
!!    Recommended value: 5.0
SMT_UPDATE_PERIOD_IN_REPLAY          2      5.0
!!
```

Simulation mode Parameters: modifications shall only be done in coordination with CGS development team

```
!!
!! (E) The max number of adu_generator tasks, i.e. the max number of different
!!    adus that can be simulated in parallel
!!    Range: positive'range
!!    Recommended value: 100
ADU_GENERATOR_POOL.MAX_NO_OF_AGENTS       1       100
!!
```

System Parameters: modifications shall only be done in coordination with CGS development team

```
!!
!! (E) The number of message_agent tasks, i.e. the max number of requests from
!! TES_RPI or TES_API being served in TES_Core in parallel
!!    Range: natural'range
!!    Recommended value: 10
```

REQUEST_FETCHER.NO_OF_MESSAGE_AGENTS 1 10
!!

9 TEST EVALUATION

9.1 General

CGS allows for the evaluation of data stored in the Test Result Database (TRDB) during a running test session or after having executed a test session with the Unit Under Test (UUT) or with simulated parts of it.
The data archived often needs further evaluation in an off–line session, especially to verify whether data generated by the UUT is in the required margins or to analyse non–nominal situations during a test.

In the beginning of a TEV session the user must define the data area in the Test Result Database (TRDB) to be evaluated. This means providing the name of test session(s) for which data shall be evaluate and eventually a time frame.

During the evaluation work the user frequently produces output files, held by default under the user's account, these output files produced are the so–called **evaluation result files** and setup definition files.
The user is also allowed to specify an "Evaluation Session", under which he can store the evaluation result files in the TRDB.

Finally, the user may convert Data Set and Data Listing into a file in Excel format, in order to proceed the evaluation with Excel. (see Chapter 9.5.1 File Handling).

The following basics steps describe the general proceeding to be performed in every TEV session:

- Select the test session(s) to evaluate
- Select or create a test evaluation session (optional)
- Select a Configuration Unit of the Mission Database (CCU) containing the test data
- Select the appropriate TEV tool for data evaluation
- Store the evaluation results locally (optional) (the evaluation results are per default stored in the Test Result Database)
- Convert Data Set and Data Listing in Excel format, if the user wants to proceed to an evaluation through the Excel Tool.

Note that TEV provides an On–Line_Help facility giving access to the TEV user manual.

The On–line Help provided by TEV is based on the help features provided by the OpenWindows environment. The Help mechanism is supplied to the user via two different ways :

- From the main menu, a Help button allows the user to access the user manual of TEV inside a Help Viewer window (OpenWindows application). The user can navigate inside the TEV user manual by using the features provided by the application (Cfr. OpenWindows manual),
- From the Help key on the keyboard, the user can request information about any TEV window. When this key is pressed, a Help "push pin window" is displayed containing textual information about the window item pointed by the cursor. From within this Help window, a "More" button allows the access to the TEV user manual via the Help Viewer window.

These two mechanisms are shown in the next figure (Figure 9–1).

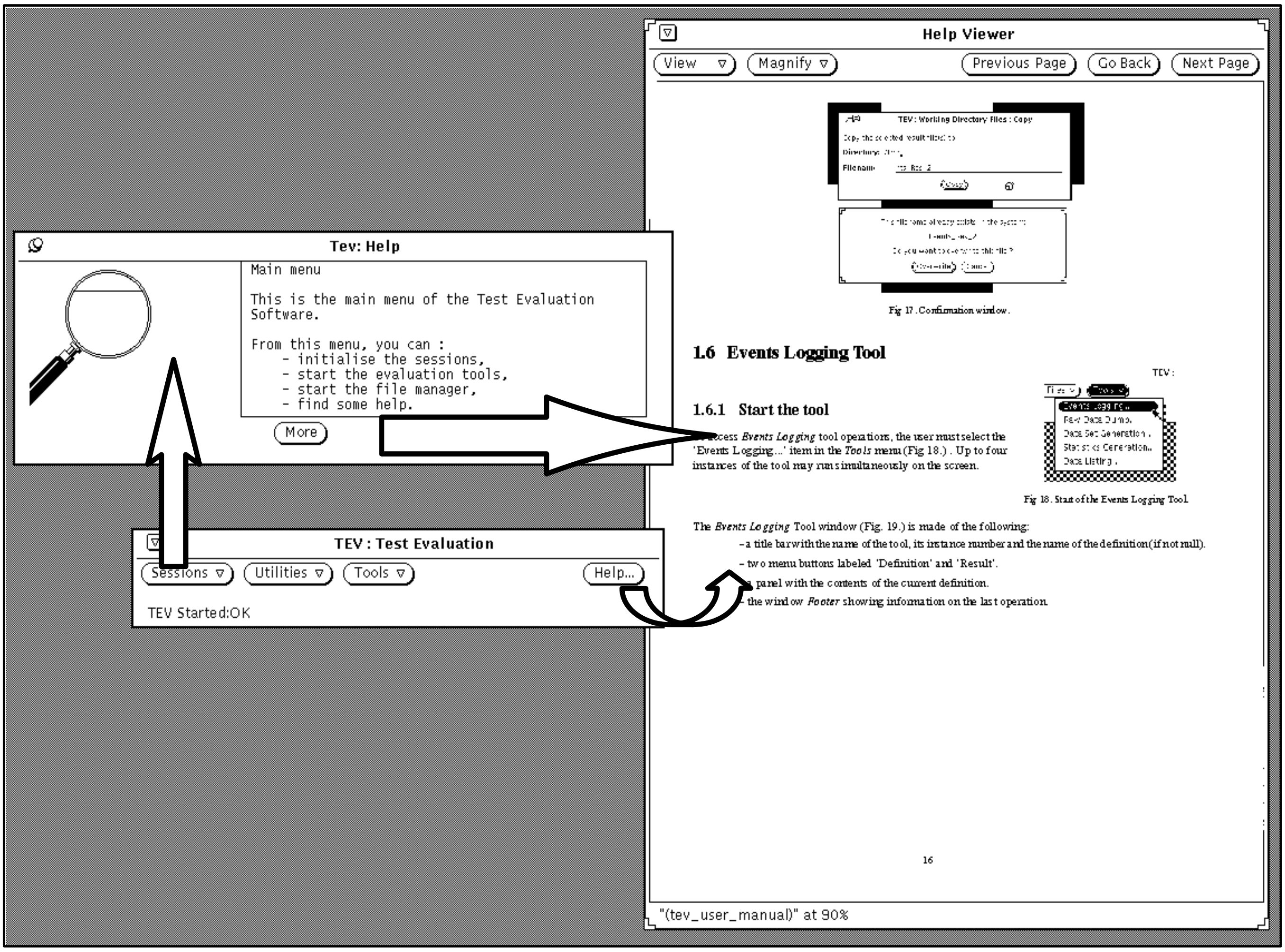


Figure 9–1 : *On–Line Help mechanism*

TEV can be started under two different initialisation modes : EGSE mode and OFFICE mode. In OFFICE mode, TEV is responsible for the startup and shutdown of DBS processes. To invoke OFFICE mode, TEV should be started at the command line with the parameter –s. By default TEV is started in EGSE mode. Under OFFICE mode, if others evaluation users are connected then the Central DBS processes will not be killed. If Central DBS is already running and TEV is started in OFFICE mode, then TEV starts normally with no error message. TEV may also be started in BATCH mode. When started in BATCH mode the TEV initialisation mode is EGSE. There is no possibility to start a BATCH TEV in OFFICE mode.

TEV starting conditions in OFFICE mode are as follows:

- the user must be logged on the system
- the environment variable DISPLAY must be defined (DISPLAY must be set using the ***host id name***)

9.2 Using TEV in batch mode

TEV provides a batch utility through which the user may execute the logging event tool, the raw data dump tool and the data set generation tool. The TEV execution conditions are defined in the input parameters of the batch command. Each batch process started executes one batch command.

The user enters the batch command from the command line. This invokes the start script which recognises that batch TEV is to be started. The input parameters are passed to the start procedure. The TEV configuration file is read to determine whether the archived file list is to be provided by DBS or TDCS and whether the data set generation is to be performed internally or externally.

It is only possible to run one instance of TEV at a time on a workstation. Multiple instances of interactive TEV and batch TEV may be run on different stations.

Inputs :

Four inputs are required to invoke batch TEV :

- a tool identifier to specify the tool to be executed.
- a definition file containing the definition to be executed.
- a session file containing the execution sessions with which TEV is to be initialized and the CCU information.
- the name of the result file where the result is to be written.

The inputs may be supplied in any order but the following syntax should be used :

```
start_tev  –to <tool_name> \
           –se <session_file_name> \
           –de <definition_file_name> \
           –re <result_file_name>
```

where **session_file_name, definition_file_name** and **result_file_name** do not include complete pathnames but are only filenames.

Tool Name :

The tool name specifies the TEV tool which is to be executed and must be one of the following :

- **LOG_EVENT** for executing the log event tool
- **RAW_DATA** for executing the raw data dump tool
- **DATA_SET** for executing the data set tool

Session File :

The session file contains the sessions with which TEV is to be initialised and the CCU information. It is an ascii file and must exist in $VICOS_TEV_WD/SESSIONS. It must be of the following format:

```
SESSION        EXEC_SESSION_01
SESSION        EXEC_SESSION_02
SESSION        EXEC_SESSION_n
SESSION        DEFAULT_TEST_SESSION
```

Optional :

```
CCU_CONFIG_NAME        Valid name of an existing CCU in MPS.
CCU_VERSION            Valid version number of CCU in MPS.
CCU_PATHNAME           Valid pathname of an existing CCU in MPS.
```

The following old format is also supported and can be also mixed with the new one :

```
SESSION:              EXEC_SESSION_01
```

SESSION: EXEC_SESSION_02
SESSION: EXEC_SESSION_n
SESSION: DEFAULT_TEST_SESSION
CCU_CONFIG_NAME: Valid name of an existing CCU in MPS.
CCU_VERSION: Valid version number of CCU in MPS.
CCU_PATHNAME: Valid pathname of an existing CCU in MPS.

This file may contain a maximum of 10 execution session names plus the *default_test_session*. The *default_test_session* may appear at any position in the list of sessions.

The sessions with which TEV is initialised will be allocated to the batch user for the duration of the job.

The mission name, element configuration name and system tree version number are obtained by TEV from the session(s).

If any or all of the ccu configuration name, the ccu version or the ccu pathname are missing or wrong then these are obtained by TEV from the session(s). If the sessions(s) have different CCUs, TEV takes the one from the most recent session.

It is not necessary to write the identifying token if the data is not supplied through the session file.

Definition file :

The definition file contains the definition to be executed and must be in TEV internal definition file format. The file is created using the menu ***”Definition:Save”*** in the appropriate tool in TEV interactive.If the definition is saved while ***”Select Initial Time Frame”*** is selected, the definition contains the information ”initial Time Frame” of the session(s) and will be adapted to the sessions(s) whenever these are produced. Otherwise the ***”Selected Time Frame”*** is part of the definition.
This file may be also produced using either the TEV_API or through a TEV_SAS. The definition file must be of the same type as the tool specified and must exist in the appropriate definition directory ie.
– $VICOS_TEV_WD/DEFINITIONS/EVENT_LIST or
– $VICOS_TEV_WD/DEFINITIONS/RAW_DATA_DUMP or
– $VICOS_TEV_WD/DEFINITIONS/DATA_SET.

Result file :

Once the definition has been executed the result is written in the result file which is created in the appropriate directory :
– $VICOS_TEV_WD/RESULTS/EVENT_LIST or
– $VICOS_TEV_WD/RESULTS/RAW_DATA_DUMP or
– $VICOS_TEV_WD/RESULTS/DATA_SET.

The created result file is in the appropriate ADT evaluation result file format ie :
– ADT_EVENT_RESULT or
– ADT_PACKET_RESULT or
– ADT_DATA_SET.

The result may be displayed through interactive TEV or accessed in the required manner through a TEV_SAS.

Note: In batch TEV, confirmation to overwrite an existing result file is not possible.
Existing result files will be overwritten by batch TEV.

Error Handling :

All errors will be logged to the CGSI and also written to stdout. Errors may be generated either in the batch TEV driver during the command checking or in the evaluating objects. Errors will be logged if input parameters are missing.

All messages will be written to stdout. Successful completion will be written to stdout.

To log messages and errors in a file the user should redirect the output using standard unix commands.

9.3 Test Evaluation Preparation

Test Evaluation Software starts by displaying the window in Figure 9–2 that contains the main menu buttons. These buttons activate the TEV operations for the SESSIONS, UTILITIES and the EVALUATION TOOLS.

The **Sessions** menu button allows the user to :

- select an Evaluation session to store or retrieve result files,
- to delete a given Evaluation session,
- to select Execution sessions and the CCU,
- to delete a given Execution session,
- to archive/retrieve Execution or Evaluation sessions to/from the Final Archive Medium and to delete on–line data,
- to export and import sessions to/from the Final Archive Medium

The **Utilities** button allows the user to :

- start the file manager, for handling of evaluation result files and external files in the working directory.
- start the data sets merger utility to merge two data sets.
- start the data set/events listing merger
- start the job queue utility to display and cancel FA jobs and to display and cancel print jobs.
- start an SAS

The **Tools** button allows the user to start the Test Evaluation tools.

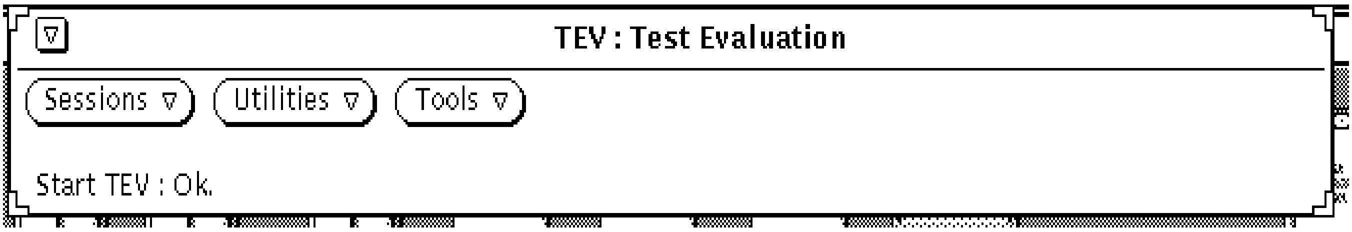


Figure 9–2 : *The TEV main window with its menu buttons*

The message **Start TEV : OK** is the status message reported to the user in the base window of the application. This information line has two parts separated by a colon(:): the operation executed and its returned status. The status may be Ok or an error message. For each tool main window in TEV, context driven messages will appear in the footer.

When starting TEV, some errors may arise, like a DBS connection problem, missing variables in the environment.... For more information about the TEV error report service refer to Appendix D.

The user can access the TEV functions via the buttons **Sessions**, **Utilities** and **Tools** in the main window.

Note that the main window buttons (and some other buttons) work in two different ways :

Selecting explicitely a function from a pop–up menu

- Move the mouse pointer on the button.
- Press and hold the right mouse button. The pop–up menu is displayed.
- Move the mouse pointer up and down the pop–up menu still holding the mouse button until you find the desired function. The function becomes highlighted.
- Release the mouse button. The desired function is activated (e.g. a new window is opened).

Activating the default function from a pop–up menu

- Move the mouse pointer on the button.
- Click on it with the left mouse button. The default function in the pop–up menu is activated.

9.3.1 Evaluation Session Definition

An Evaluation Session has to be specified if the user wants to store the results of their data evaluation activities (i.e. the Evaluation Result files) in the TRDB.
The Test Evaluation can be performed without specifying an Evaluation Session. Per default the result files are stored in a so-called ’Working directory’ in the user’s own UNIX directory structure : users_home/*wd/tev/RESULTS*. The user has no possibility to set–up a mode, which would implicitely store all the result files in the current selected evaluation session. The user shall explicitely use the **Store** button from the **File Handling** Utilities.

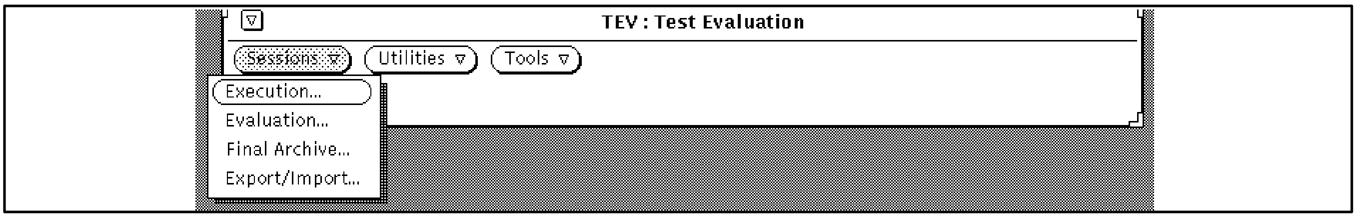


Figure 9–3 : *The **Sessions** menu*

To access Evaluation Session operations, the user must select **Evaluation...** item in the *sessions* menu (Figure 9–3)

The **TEV : Evaluation Session** window (Figure 9–4) appears:

Figure 9–4 : *Initialize Evaluation Session window*

List existing Evaluation Sessions

- Select a time frame to specify the evaluation sessions to be listed.
- Type a name substring (containing wildcards (*)) as filter for the session names to be listed.
- Select the order in which the sessions shall be displayed (chronological or alphabetical).
- Click on the **List** button, to update the list of sessions according to the selection criteria.

By default the time frame and name substring will select all entries. This means that when evaluation sessions exist, there are by default all listed. The order by default is chronological. Note that the selection criteria are combined by a logical AND.

The user can reuse and update an existing Evaluation Session :

Select an Evaluation session

- Select a time frame to specify the evaluation sessions to be listed.
- Type a name substring (containing wildcards (*)) for selection of sessions.
- Select the order in which the sessions shall be displayed (chronological or alphabetical).
- Click on the **List** button.
- Move the mouse into the Evaluation Sessions list.
- Scroll the list by using the scrollbar if necessary.
- Click on the desired name.
- Click on the **Accept** button.

When a session name is selected in the list, any previous selection will be automatically deselected because the list accepts zero or one selection.
When a session name is selected in the list, its name is copied to the *Session Name* field and its associated information is also shown. None of these fields are editable.

Note that once the user has selected a session in the list, the left–hand side button has for label ”Accept”.

The user may see the sizes of an evaluation session for each evaluation data types : for this select the session in the list and click on the **”Session Size”** button.

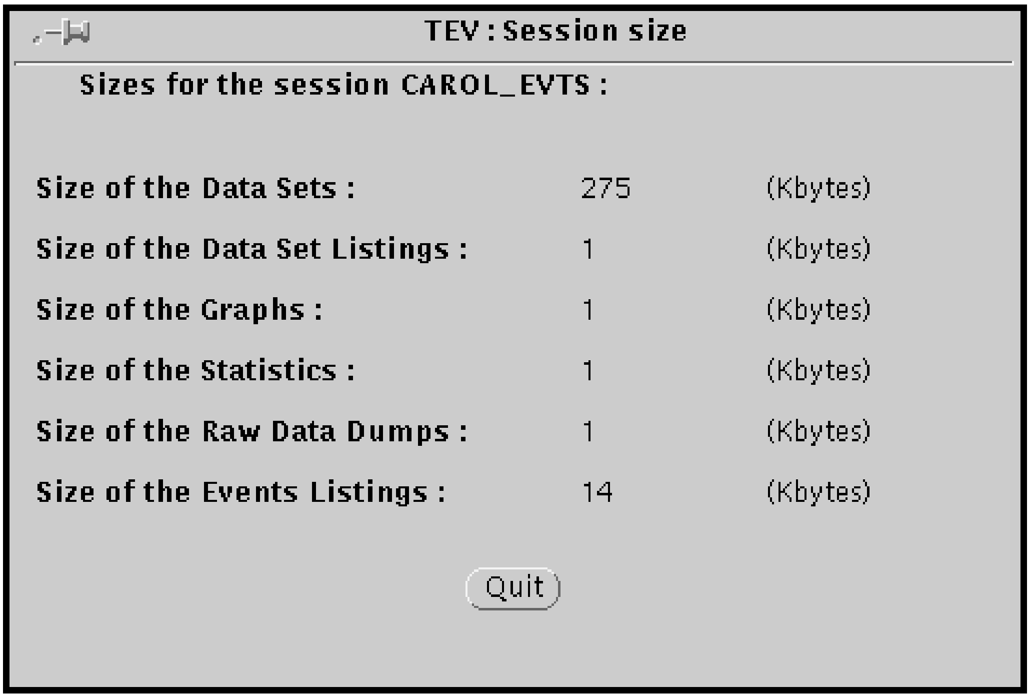


Figure 9–5 : *Evaluation Session size window*

For each TEV evaluation data types (Data Sets, Data Sets Listings, Graphs and Statistics, Raw Data Dumps and Events Listings) DBS may have created a directory structure. Usually the size is ”1” indicates the presence of an empty directory.

Knowing the sizes of a session may be important before exporting or archiving on the MO device.

The user can create a new evaluation session. To be in creation mode, no session shall be selected into the list. In this case, the left–hand side button has for label ”Create”.

Create an Evaluation session

- Deselect any previously selected session in the list.
- Type the name of a session (up to 20 characters) in the **Session Name:** text field.
- Type the purpose of the session (max. 80 characters) in the **Purpose:** text field.
- Press the **Create** button.

The new session appears in the list of evaluation sessions. All the information fields others than name and purpose are set–up by TEV, and not editable.

When the user either creates or accepts a session this session is allocated to him. This prevents the session from being deleted, archived or exported by another user. The allocation is cleared when the user disconnects ie quits TEV or creates or accepts another session. Other users may allocate the same session. A SESSION_IS_USED error will be reported if a user tries to delete, archive or export a session which is not **exclusively** allocated to him.

If there is no error when accepting the selection, the Evaluation Session window is iconised. An error may be caused by an invalid session name, a missing purpose, a problem accessing the Test Result Database...

9.3.2 Execution Session Initialization

The user must select a set of execution sessions from the TRDB, in order to generate Raw Data Dumps or Data Sets. It is not necessary to select sessions to load some already existing Raw Data Dumps, evaluate existing Data Sets or perform some merge. Nevertherless, it is at least necessary to select a CCU in order to access MDB items (for example to generate a Definition). A CCU is automatically selected when one session or more are selected.
The user can also select explicitely a CCU and/or store a default CCU, which will be automatically selected as default.
The selected execution session(s) as well as the selected time frame identify the set of data which can be dumped and for which Datasets can be generated.

The test execution sessions may be selected according to time frame, name, state, location and mode selection criteria.

The Execution Sessions Initialization operations are :

- the selection of up to ten execution sessions from which TRDB data are extracted
- the selection of a CCU

Note that the **Select Execution Sessions** *window (Figure 9–6) may be displayed at any time, changes will only be accepted if no tool is running.*

To access Execution Sessions operations, the user must select the ’Execution...’ item in the *Session* menu (Figure 9–3) .

The **TEV : Select Execution Sessions** window (Figure 9–6) appears :

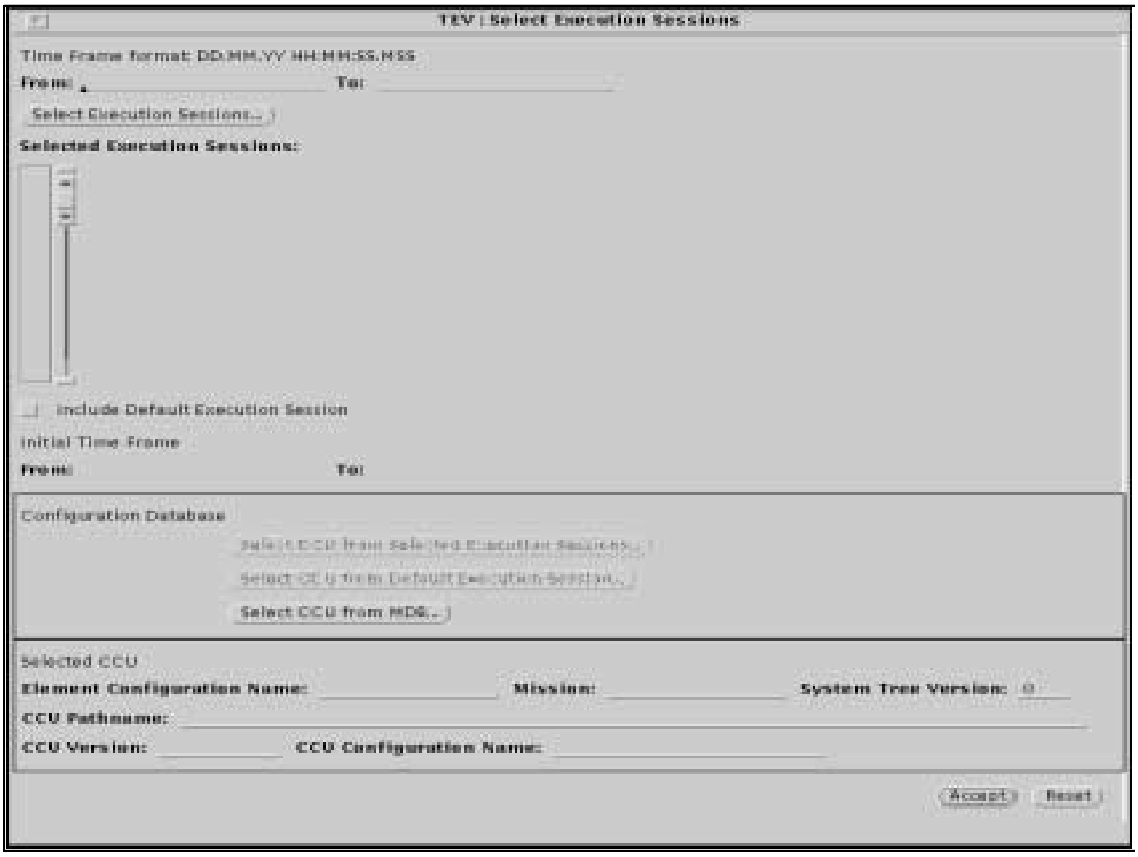


Figure 9–6 : *TEV: Select Execution Sessions window*

The **TEV : Select Execution Sessions** window is split into 3 areas :

- the first area concerns the selection of execution sessions
- the second area concerns the explicit selection of a CCU
- the third area displays the information about the currently selected CCU and is read only

9.3.2.1 Select execution sessions :

General overview of the procedure to select executions sessions :

Select Execution Sessions

- Press the **Select Execution Sessions...** button.
- Wait until the **TEV : Execution Session List** window pops up.(see the description : ***The window TEV: Execution Session List functionalities*** below).
- Select criteria for listing the execution sessions.
- Click on the **Apply** button to validate the selection criteria and update the list of execution sessions available for selection.
- Select the desired Executions Sessions from the **TEV: Execution Session in the Time Frame** list.
- Press the **Accept** button. The selected sessions are now displayed in the **TEV : Select Execution Sessions** window.

*In the area **Seleted CCU** appears the CCU which is selected. A warning message will be displayed if the sessions selected were created with different CCUs. In this case the CCU of the newest Execution Session will be selected and the evaluation is limited to those parameters known from the selected CCU. The newest Execution Session is the session with the newest begin date.*

- Click on the **Include Default Execution Session** check box in the window **TEV: Select Execution Sessions**, if the data from the default test session has to be included.

The Default Execution session contains the test logs of those test execution sessions which were started without defining an execution session by name. The default session may be selected together with other sessions in the list or alone.

- Specify a Time Frame to restrict the data to be evaluated (if no Time Frame is specified, then all the data will be used for evaluation).

The window *TEV : Execution Session List* functionalities :

A first area presents a set of selection criteria to reduce the list of execution sessions to be presented for selection. The selection criteria are combined with a logical AND. A click on the button **Apply** validates the selection criteria and displays the list of execution sessions accordingly.

The second area is the list of execution sessions presented for selection.
The execution sessions list contains the session name, it’s associated time frame and the session state, mode and location. Following abbreviations are used:

• Session State:	Closed	**CL**
	Open	**OP**
	Aborted	**AB**
• Session Mode:	Normal	**NM**
	Simulation	**SM**
	Replay	**RP**
	Any Mode	**AM**
• Session Location:	On Line	**OL**
	Final Archive	**FA**
	On Line & Final Archive	**OL/FA** (i.e partially or totally re-

trieved)

The user may select up to ten sessions in this list.
The user may delete the selected sessions. A confirmation window will be displayed for each of the sessions.

Restrictions on deletion :
– The user must have the privilege to delete a test execution session. The privileges affected to one user are set–up through the $DBS_HOME/config/dbs_privilege_file.def file and the file $CGS_HOME/config/USER_PROFILES.
– Sessions cannot be deleted if they are allocated to another user (another TEV user has already selected these sessions)
– Sessions cannot be deleted if TEV tools are running
– the DEFAULT_TEST_SESSION cannot be deleted through TEV

The user may see the sizes of a session for each TRDB data types : for this select a session in the list (when several sessions are selected TEV will consider the last selected session) and click on the **”Session Size”** button.
For each TRDB data types (Archive files, Events, Engineering Values Logbook and configuration file) DBS may have created a directory structure. Usually the size is ”1” indicates the presence of an empty directory.

Knowing the sizes of a session may be important before exporting or archiving on the MO device.The user shall know that in this case the events table are translated into unix files before exporting or archiving.Only the number of events is then relevant to calculate the needed space.

Once the user has selected sessions in this list, a click on the button **Accept** will accept these sessions as selected, the window **TEV :Execution Session List** disappears and the selected sessions appear in the list **Selected Execution Sessions** in the window **TEV:Select Execution Sessions.**

9.3.2.2 The different functionalities to select a CCU

In some cases, the user shall select explicitely a CCU :

- when the default selected CCU (the one from the newest session) do no fit the actual needs
- when the CCU to be selected is one from the default test session
- when only a CCU, and no sessions, shall be selected

To select the CCU from a particular session, the user clicks on the button **Select CCU from Selected Execution Sessions ....**

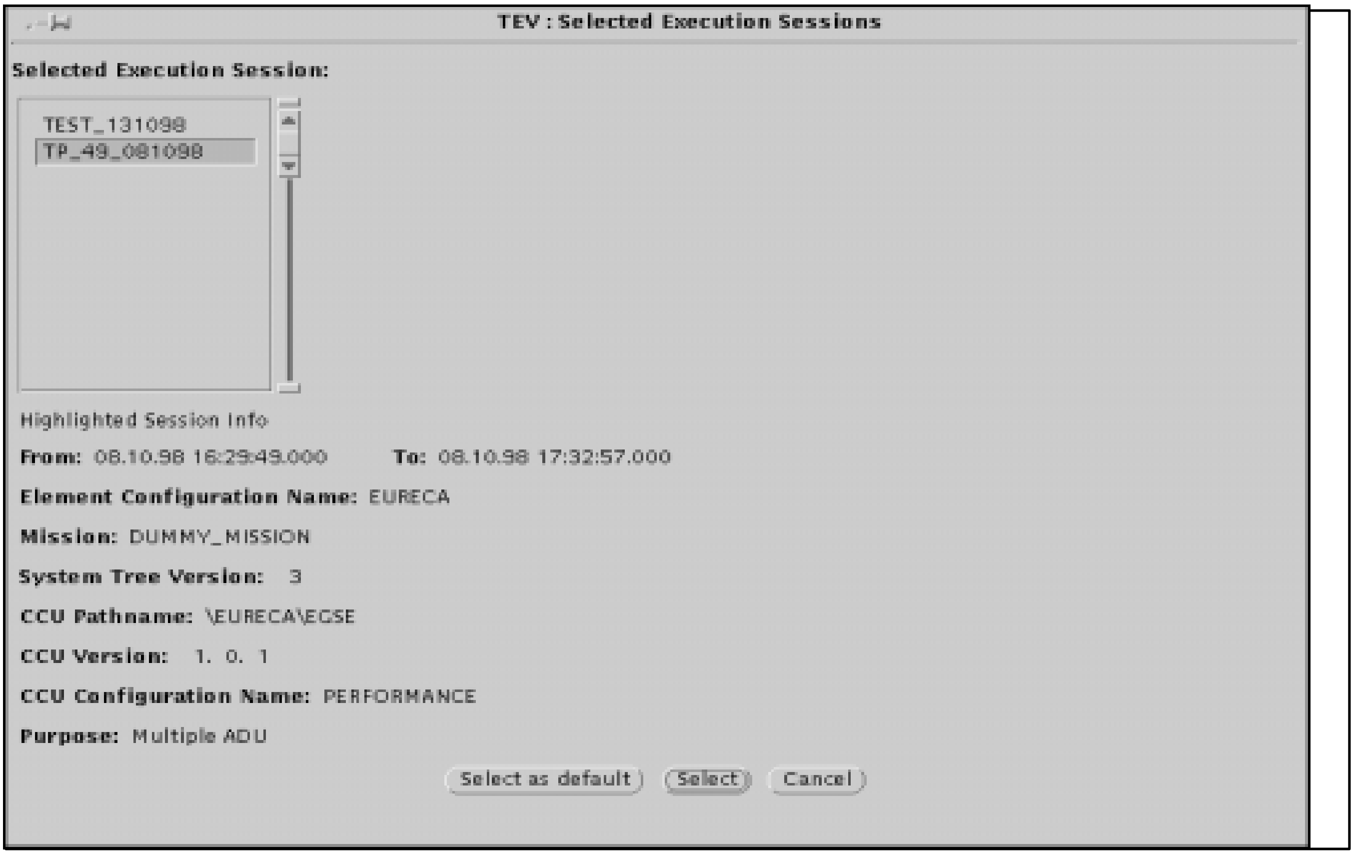


Figure 9–7 : *TEV : Selected Execution Sessions window*

Select a CCU from a selected session

- Click on the button **Select CCU from Selected Execution Sessions**. The window **TEV : Selected Execution Sessions** appears. (Figure 9–7)
- Click on the session in the list for which the CCU information shall be displayed.
- Click on the **Store as Default** button. This CCU will be always selected by default if no session are selected. This information will be kept over after quitting TEV, for the same Unix user.

The Default CCU is the one which is selected, even if no sessions are selected.
*This CCU will also appear as default in the window **TEV:CCU List in MDB.** The Default CCU information is kept over after quitting TEV for the same Unix user*

The text fields below the Highlighted Session Info heading are read–only and contain the informations about the selected session in the list.

- Click on the **Select** button to select the CCU

*In the area **TEV: Select Execution Sessions : Seleted CCU** appears the CCU which is selected.*

To select the CCU from the Mission Database, the user clicks on the button **Select CCU from MDB ....**

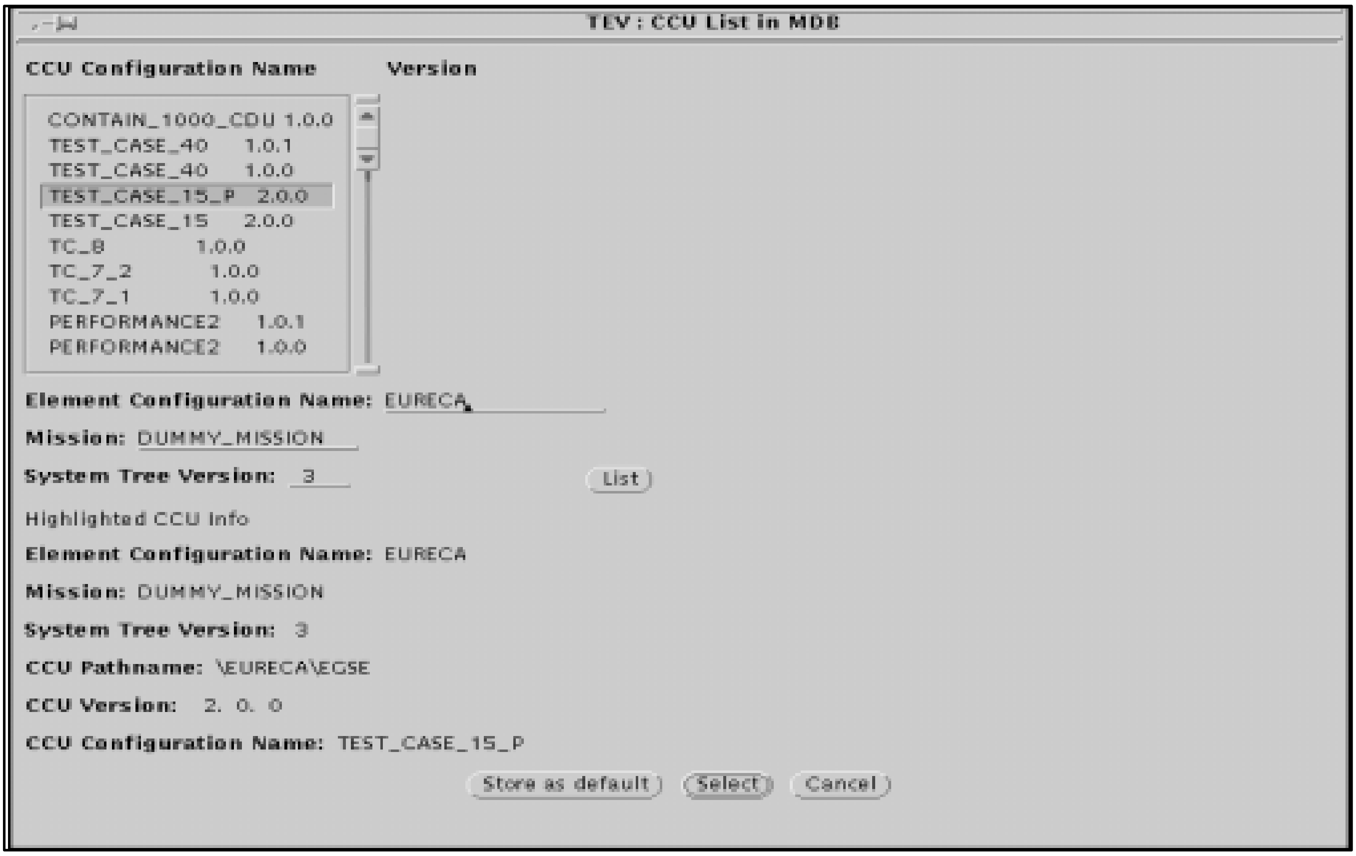


Figure 9–8 : *TEV : CCU list in MDB window*

Select the CCU from the MDB

- Press the **Select CCU from MDB ...** button. The window **TEV: CCU list in MDB** appears.(Figure 9–8).

The initial CCU list is obtained for the element configuration, mission and system tree version from the default CCU. If no default CCU was previously stored, the list and fields are empty.

- Enter the desired element configuration, mission and system tree version, then press the **List** button.
- The selected CCU can be stored as default : click on the **Store as Default** button**.**

The Default CCU is the one which is selected, even if no sessions are selected.
The Default CCU information is kept over after quitting TEV for the same Unix user

- Click on the **Select** button.
The CCU is selected and appears in the area **Selected CCU** in the **TEV:Select Execution Sessions** window.

To select the CCU from the Default Test Session, the user clicks on the button **Select CCU from Default Execution Sessions ....**

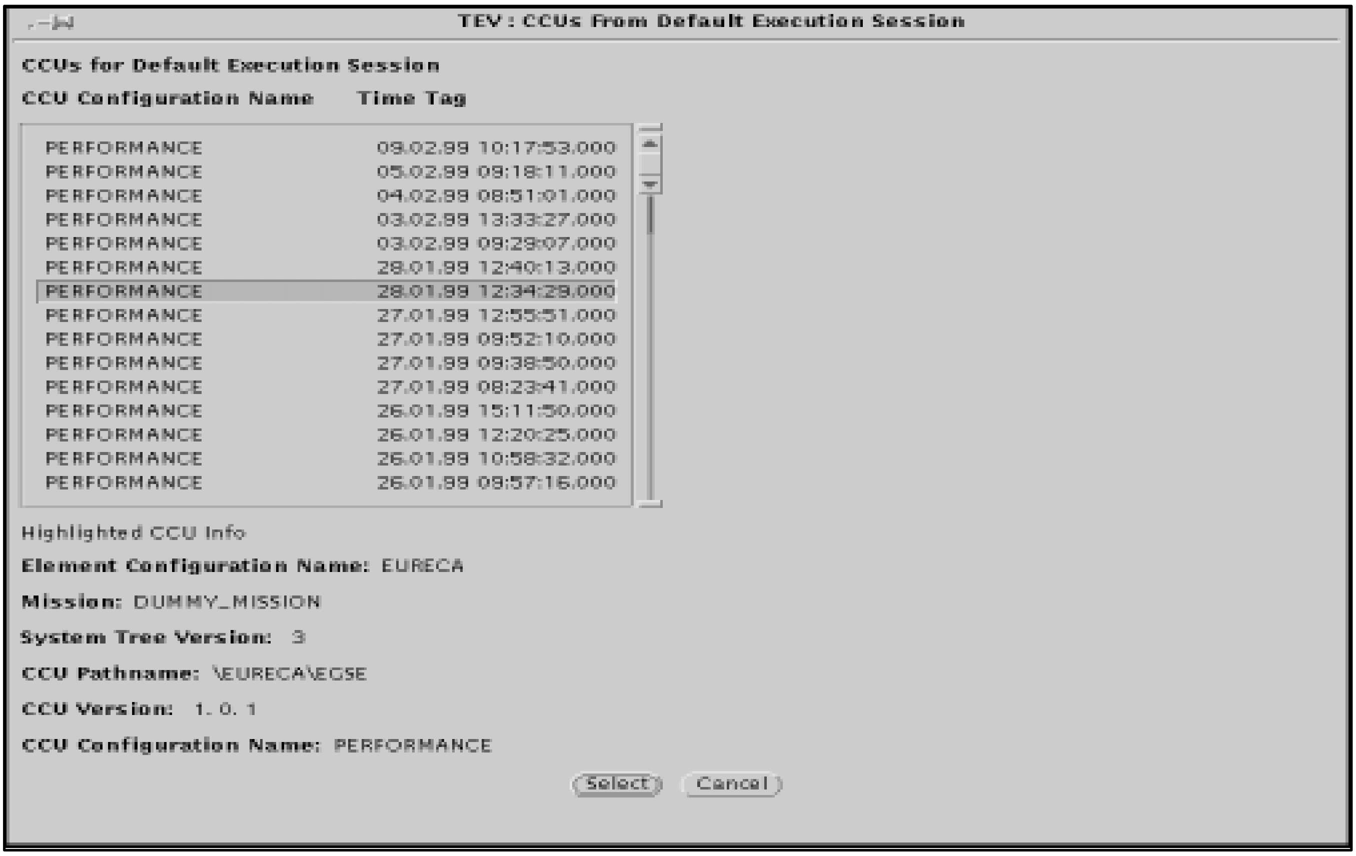


Figure 9–9 : *TEV: CCUs From Default Execution Session window*

Select the CCU from the list of CCUs used by the Default test session

- Press the **Select CCU from Default Execution Session...** button. The window **TEV: CCU list in MDB** appears.(Figure 9–9).
- Click in the list to obtain the information about a CCU
- Click on the **Select** button.
  The CCU is selected and appears in the area **Selected CCU** in the **TEV:Select Execution Sessions** window.

9.3.2.3 Validate the selections :

Validate the selection of execution sessions and CCU

- Press the **Accept** button on the bottom of the **TEV : Select Execution Sessions** window. If the selection is accepted by TEV, the window is iconised immediately.

A compliance check is performed to assure that the selected CCU exists in the configuration database. The selected sessions will be allocated on a successful accept, this prevents deletion or archiving by another

user. The allocation is cleared whenever a re–initialisation is performed (i.e a new **Accept**) or when the user quit TEV.
An **Accept** cannot be performed if another TEV tool is open.

The Reset button reverts to the state of the last successful **Accept**.

- *An* ***Accept*** *will be performed even if the sessions selected are already selected and in use by another TEV user. This means that none of these users will be able to perform deletion or archiving for these sessions.*

- *The message :* **Warning : CCU with no proper consistency status selected from MDB** *don't prevent the user to evaluate the selected session(s), as far as those information needed from the MDB are consistent.*

9.3.3 The Final Archive Tool

This tool allows the TEV user to move TRDB areas to the FA medium (*Archiving*) or copy temporary FA data to the TRDB (*Retrieving*). Two types of location are then possible for data : On–line data are data located on the DBS central disk, Archived data are data located on the FA medium.

The TEV **Final Archive Tool** will generate requests to the FA SAS, part of DBS which is driving the Final Archive Medium (optical disks). The FA SAS generates then through its own User Interface questions to the user. It is very important to note that the TEV **Final Archive Tool** can only work in parallel with the FA SAS tool. For more information on the FA SAS, see Chapter 10.2.

The Final Archive Tool is accessed through the 'Final Archive...' item in the *Sessions* menu (Figure 9–3). Once the **TEV : Final Archive Tool** window (Figure 9–10) has been displayed, the user is able to archive and retrieve Execution and Evaluation Sessions.

- *Archiving and Retrieving operations are controlled through privileges mechanisms. The privileges affected to one user are set–up through the $DBS_HOME/config/dbs_privilege_file.def file and the file $CGS_HOME/config/USER_PROFILES.*

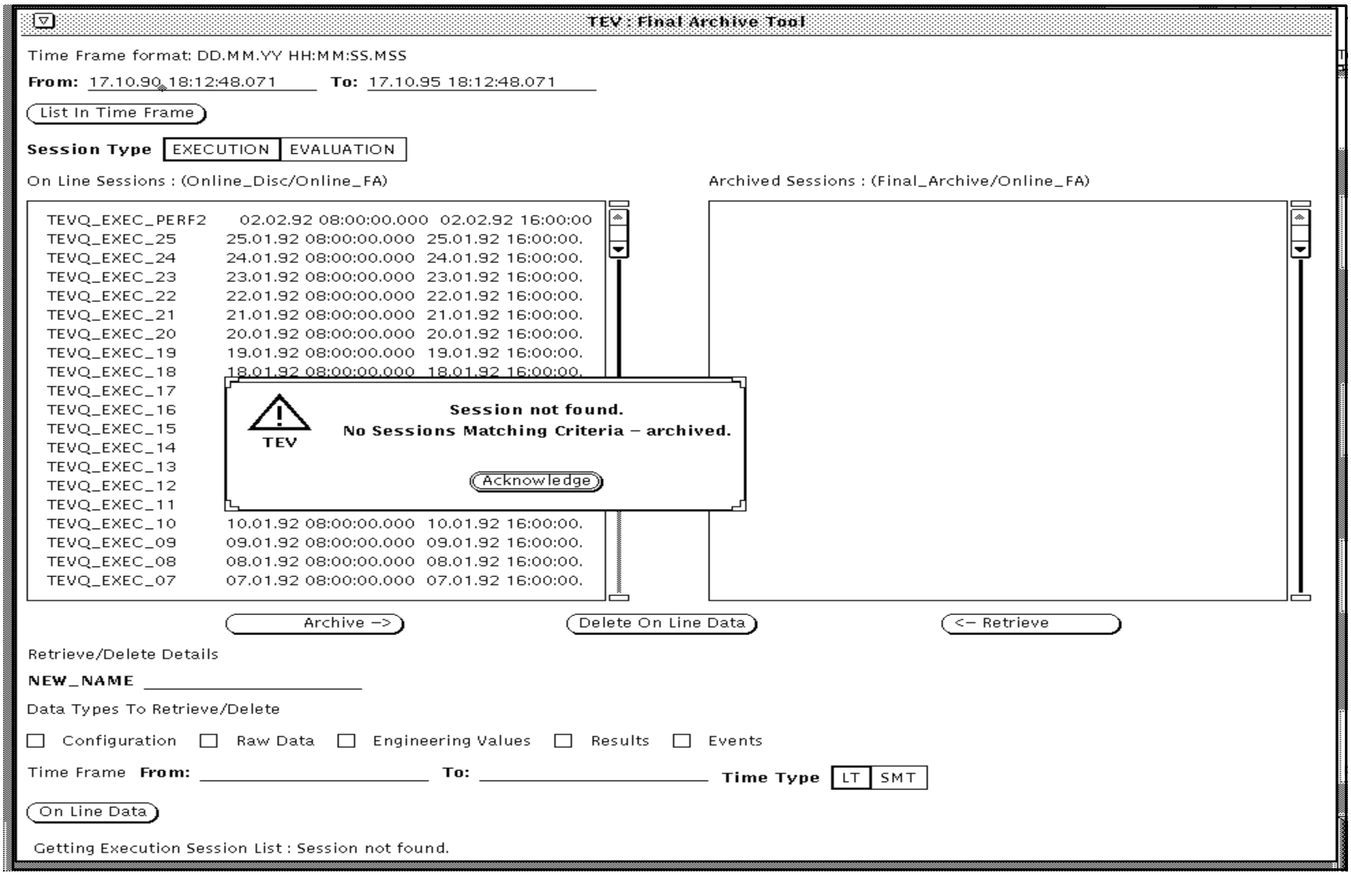


Figure 9–10 : *Final Archive Tool window*

This window is composed of the following :

- To startup the FA_SAS, the button *Start_FA_SAS* is available. It startsup the process on the DB Server node.
- To request the FA_SAS to display the FA device status, the button *Display_FA_SAS* is available.
- a *Time Frame* field to restrict the sessions presented in the list. This time frame is initially set to the current date and time.

Because of this ”NULL” initial set–up of the Time Frame, the lists appear initially empty.

- a *List in Time Frame* button to update the list of On Line Sessions and the list of Archived Sessions according to the Time Frame selected.
- a check box to specify the *Session Type* : Evaluation or Execution to display into the lists.
- an *On–Line Sessions* list with the list of sessions which may be archived (sessions on the DBS central disk) and sessions partially on–line (archived sessions from which data have been retrieved).
- an *Archived Sessions* list with the list of sessions which may be retrieved i.e.from which the data are archived on the Final Archive Medium.
- two buttons: *Archive* and *Retrieve* to start the two operations.
- a *New Name* field used to specify the new name of a retrieved session,

- a set of check–boxes used to specify which data types have to be retrieved or deleted.
- a second *Time Frame* field to restrict the data to retrieve.
- an *On–Line Data* button to check the location of the data for one selected session.

*The message **Session not found** is displayed when the tool could not find a list of On–line sessions or could not find a list of Archived sessions inside the given time frame.*

Note that sessions partially or totally retrieved appear in the two lists.

A single selection only is possible from the lists i.e. selecting a name in one of the lists has the effect of deselecting a selected name in the other list.

Archive a session

- Select an Execution Session by clicking on its name in the **On–Line Sessions** list

*At this stage, the **Retrieve** and the **Delete On Line Data** buttons are disabled as it is only possible to retrieve sessions from the list of Archived sessions and to delete On Line data of an Archived Session. The **New Name** field is disabled as it is only possible to archive a session under the same name. The **Data Types to Retrieve/ Delete** check boxes are disabled as it is not possible to archive partially a session.*

- Press the **Archive** button.

At this stage, a dialog can be requested by the FA SAS window manager, which asks complementary information for archiving (disk name, etc...). When the user do not answer to this request, the Archive operation will fail.

By pressing the *Archive* button, the operation is scheduled in the Final Archive Job Queue. On completion the session name will disappear from the on–line list and appear in the archived sessions list. A status will also appear in the frame footer. If the session is already archived or has been allocated by another user then appropriate messages will be issued and the archive will not occur.

Retrieve a session

- Select an Execution Session by clicking on its name in the **Archived Sessions** list.

*At this stage, the **Archive** button is disabled as it is not possible to archive a session selected from the list of Archived sessions.*

- Verify which data are already on–line for the session, by using the **On Line Data** button.
- Select from the **Data Types** boxes the data types to retrieve.

No data types selected defaults to retrieve all data types.

- Enter a new name for the session into the field **New Name.**

The default name is the original name.

- Select a time frame to limit the range of retrieved data and a time type to precise to which time type applies the time frame.
- Press the **Retrieve** button.

At this stage, a dialog can be requested by the FA SAS window manager, which asks complementary information for retrieving (disk name, etc...). When the user do not answer to this request, the Retrieve operation will fail.

After completion of the Retrieve operation, the session name appears into the on–line sessions list.

It may not be possible to retrieve the session as **part** of its data is already on line. By clicking on the *On Line Data* button the user is able to display a window (Figure 9–11) to show the location of each of the data types of the selected session, it’s start and end local times and it’s start and end simulated times.

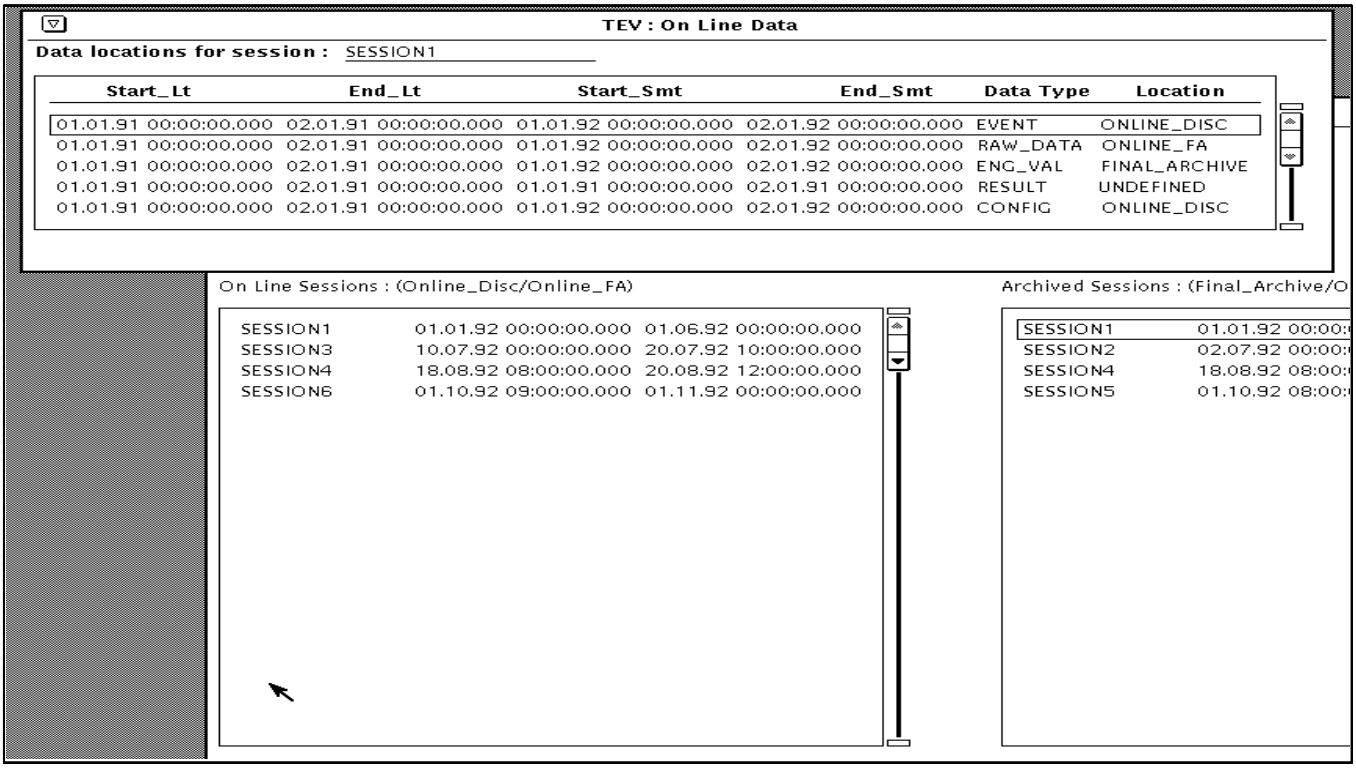


Figure 9–11 : *On–Line Data window*

Delete On–Line Data

- Select an Execution Session by clicking on its name in the **Archived Sessions** list
- Verify which data are already on–line for the session, by using the **On Line Data** button.
- Select from the **Data Types** boxes the data types to delete from the on–line location.

No data types selected defaults to delete all data types.
No account will be taken of the time frame, this means that whatever the time frame, the deletion of a selected data type will be complete.

- Press the **Delete On–Line Data** button.

The user will be asked for confirmation.

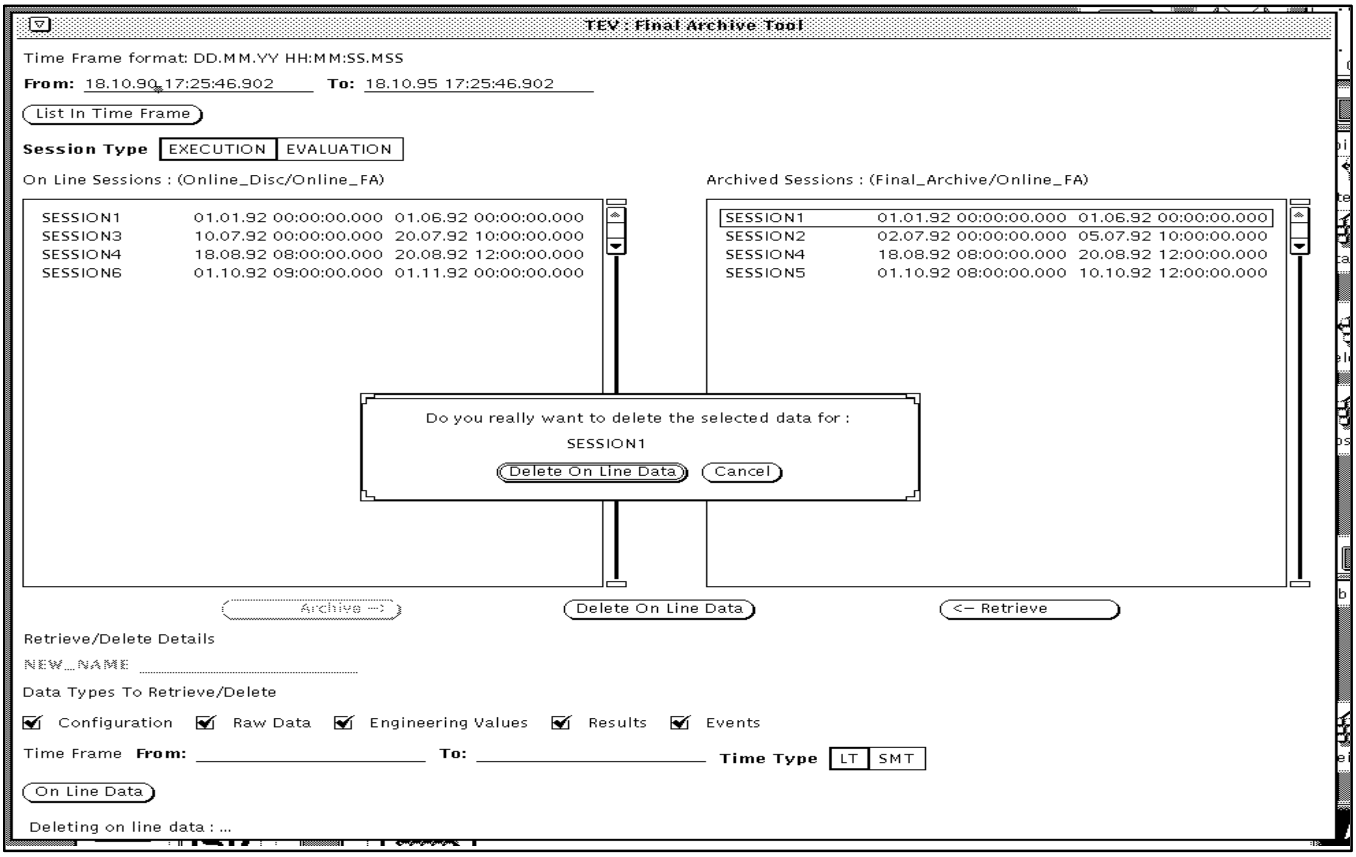


Figure 9–12 : *Delete On–Line data*

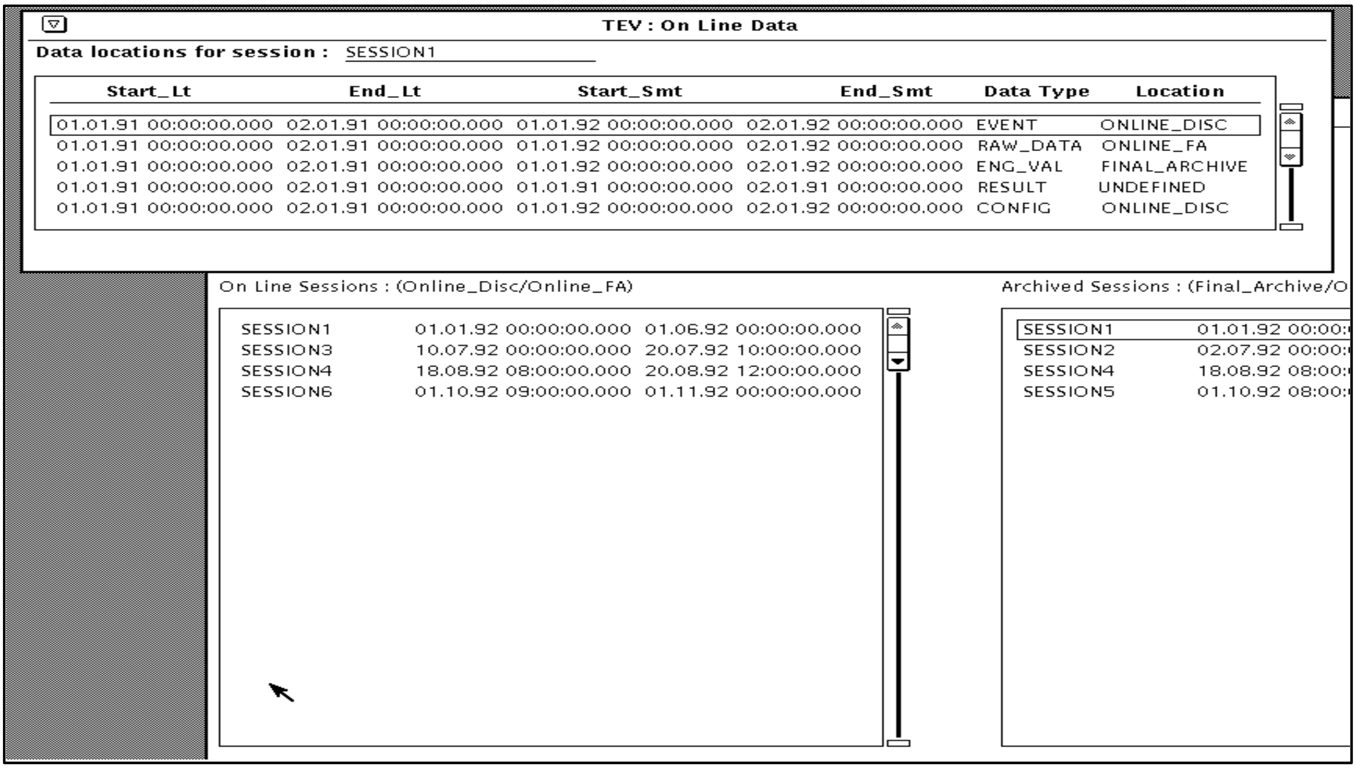


Figure 9–13 : *On–Line Data window*

9.3.4 The Export/Import Tool

Complete or partial execution sessions and complete evaluation sessions may be exported through this tool. Export means to copy the selected data from a session or complete session to the FA medium. To import is to copy a named session from the FA medium to the TRDB. Sessions imported are always considered to be complete sessions.
The TRDB keeps no references to an exported session. An exported session is a double of a session which stays unchanged into the TRDB. The disk where the exported session has been copied can be taken over and used within an other TRDB (opposed to that, an archived session is still part of the TRDB, for which references are kept).
To access the Export–Import Tool, the user must select the ’Export/Import...’ item in the *Sessions* menu (Figure 9–3).

To startup the FA_SAS, the button *Start_FA_SAS* is available. It startsup the process on the DB Server node.

To request the FA_SAS to display the FA device status, the button *Display_FA_SAS* is available.

The **TEV : Export/Import tool** works in parallel with the FA SAS tool. For more information on the FA SAS tool, refer to Chapter 10.2.

Export and Import operations are controlled through privileges mechanisms. The privileges affected to one user are set–up through the $DBS_HOME/config/dbs_privilege_file.def file and the file $CGS_HOME/config/USER_PROFILES.

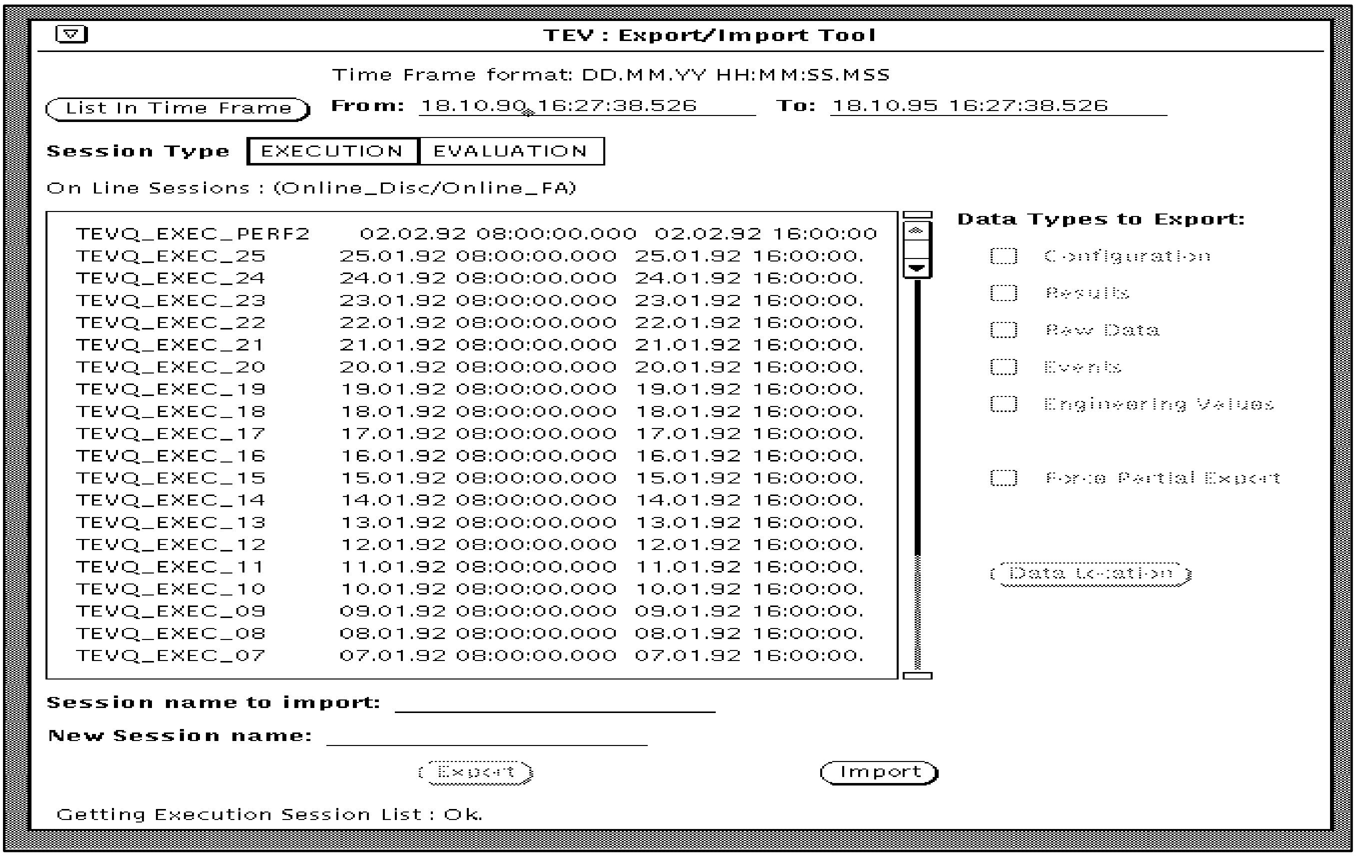


Figure 9–14 : *Export/Import Tool window*

The **Time Frame** field is initially set to the current date and time. The **List in Time Frame** button allows the user to list the On Line Sessions included into the time frame selected.

Export an evaluation session

- Select **Session Type : Evaluation**
- *The list of evaluation sessions included in the time frame is displayed.*
- Select an Evaluation Session by clicking on its name in the **On Line Sessions** list
- Enter a new name in the field **New Session name**
- Press the **Export** button.
- *At this stage, a request to export the session under the new name, or the original name by default, is sent to DBS. The FA SAS must run. It is possible that the FA SAS User Interface requires complementary information to the user. If the user does not answer, the Export operation will fail.*

Export an execution session

- Select **Session Type : Execution**
- *The list of execution sessions included in the time frame is displayed. Note that some of these sessions can be partially on–line.*
- Select an Execution Session by clicking on its name in the **On Line Sessions** list.
- Click on the **Data Location** button to verify on which location are each data types for this session.
- *Only data types on–line can be exported.*
- Select from the **Data Types** boxes the data types to export.
- *No data types selected defaults to export all data types.*
- Select **Forced Partial Export** to allow data export of only on–line data for data which is partially on line. If this option is not set and data selected for export is partially on line, the user will be warned and the export will not take place.
- Enter a new name in the field **New Session name.**
- Press the **Export** button.
- *At this stage, a request to export the session under the new name, or the original name by default, is sent to DBS. The FA SAS must run. It is possible that the FA SAS User Interface requires complementary information to the user. If the user does not answer, the Export operation will fail.*

Import a session

- Deselect any session name selected in the list
- *The **Import** button becomes accessible, the field **Session Name To Import** becomes accessible*
- Enter the name of the session to import in the field ***Session Name To Import***
- Enter the new name of the imported session in the field ***New Session Name***
- Press the **Import** button.
- *At this stage, a request to import the session is sent to DBS. The FA SAS must run. It is possible that the FA SAS User Interface requires complementary information to the user. If the user does not answer, the Import operation will fail.*

9.4 Data Evaluation Tools

9.4.1 Generalities

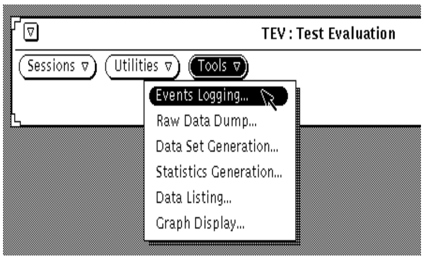


Figure 9–15 : *The different evaluation tools*

Each evaluation tool allows to extract specific data from the TRDB and format them in a specific way.
The Events Logging Tool allows to extract events and displays them in a listing.
The Raw Data Dump Tool allows to extract ADUs and GDUs packets from archive files and displays them as raw values.
The Data Set Generation Tool is an intermediate tool which does not provide a result directly accessible to the user : it allows to extract measurements from archive files or from engineering values logbooks and generate a file of engineering values, called a Data Set.
The Statistics Tool provides statistics on a set of values stored into a Data Set.
The Listing Tool provides the listing of a set of values stored into a Data Set.
The Graphics Tool provides graphics from values stored into a Data Set.

Up to four instances of each evaluation tool may run simultaneously on the screen. Each tool main window title bar contains the identification name of the tool, the instance number of the tool and the name of the current definition (if not null). Each tool main window contains a window Footer displaying information on the last operation. Most of the tools main window contains the menus *Definition* and *Result*, which are explained below.

The *Definitions* menu

The main window of every evaluation tool is based upon the current definition concept.
A definition defines, for each type of evaluation tool, criteria of selection for data and parameters for executing the evaluation. The user shall either used the NULL definition which appears by default when opening a tool or defines their own definition. The current definition is used for generation of an evaluation result. The current definition is the definition actually displayed in the main window.

The *Definition* menu provides the applicable operations on definitions : *Load, Save, Print, Print Current, Clear* and *Reset*, as shown in Figure 9–16.

Operations in the menu ending with an ellipsis (...) need a definition name (name of the file containing the definition). When these kind of operations are started, a pop–up window with a list of existing definitions (for the current tool type) is displayed for the user to select a name or enter a name directly via the keyboard.

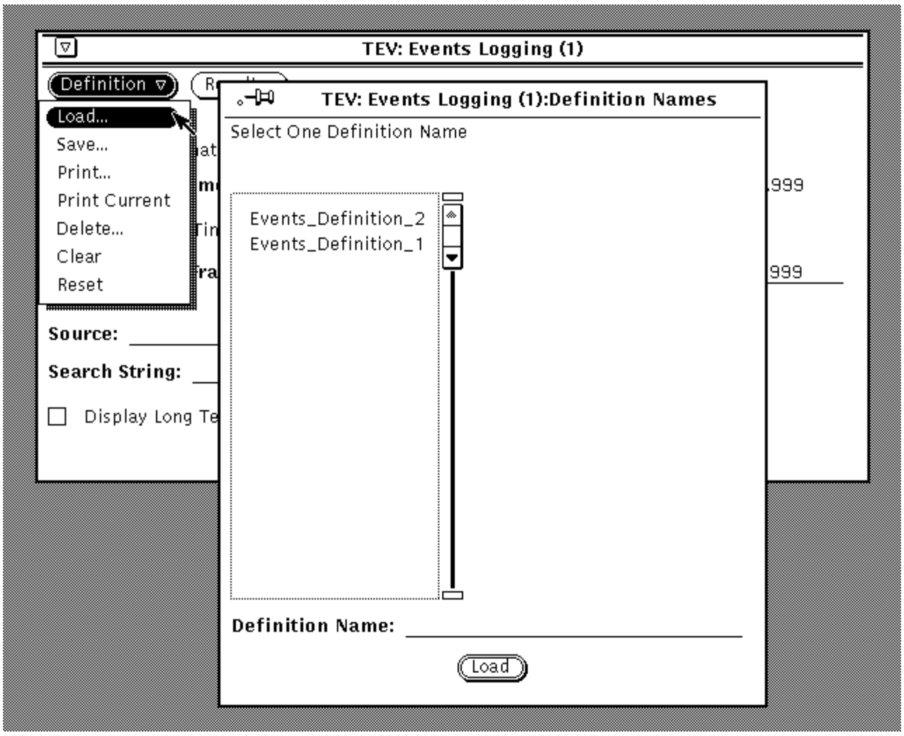


Figure 9–16 : *The Definition menu and its pop–up menu*

The *Load...* operation loads the named definition, which becomes the new current definition. The definition fields in the current evaluation tool window are overwritten, without warning. The name of the definition is displayed in the title bar. The definition file is loaded from the directory user_home*/wd/tev/DEFINITIONS/*result_type_of_the_tool.
If the short texts from two selected user events where stored in the definition, these appear in the fields : **First Selected User Event** and **Last Selected User Event**. In this case the **Selected Time Frame** fields become empty. The selected Time Frame is calculated by TEV when the user starts the **Exec&Display** process. The calculated Selected Time Frame can be different for every session containing these User Event short text.

Select a predefined definition

- Select **Definition –> Load...** from the pop–up menu.
- Select the name of the desired definition from the **Definition Names** list
- Press the **Load** button.

The *Save...* operation saves all the fields of the definition into a file.
The short texts of the two selected user events in the fields : **First Selected User Event** and **Last Selected User Event** are part of the definition and will be saved when the definition is saved. This allows the user to work with different sessions, using the same definition of a Time Frame described through User Events . These fields are saved prior to the Selected Time Frame.
The definition is first checked for errors and saved under the given name if there is no error. If the chosen name already exists, a notice box asking for confirmation of overwriting is displayed with two buttons 'Overwrite' and 'Cancel'. In case of successful completion, the name of the definition in the tool window title is modified to the new name. The saved definition still stays the current definition. The definition file is saved into the directory of the user under */wd/tev/DEFINITIONS/*result_type_of_the_tool.

Save a definition

- Tailor the definition for users needs.
- Select **Definition –> Save...** in the pop–up menu.
- Type the name in the **Definition Name ____** field.
- Press the **Save** button.

The *Print...* operation allows to print a selected definition.

The *Print Current* operation prints the current definition.

The *Clear* operation resets the current definition to the null definition. The fields are reset to default values as if the tool had just started. The definition name in the title is also reset. There is no warning given.

The *Reset* operation changes the current definition to revert to the state of the last successful operation. It can be used to come back to a correct state after user modifications have not been accepted.

The *Result* menu

This menu offers the operations on Evaluation Result Files :

The *Load...* operation reads a result file and displays it into a specific window. The title of this window contains the tool reference, its instance number and the result file name. The result file can be loaded either from the directory user_home*/wd/tev/RESULTS/*result_type_of_the_tool (*source:Working Directory*), either from the current open Evaluation Session (*source:Test Result Data Base*).

The *Save...* operation saves the current result into the directory user_home*/wd/tev/RESULTS/*result_type_of_the_tool. (To save a result into the current open Evaluation Session, use the *Store* operation from the File Manager tool).

The *Print File* operation prints a saved result, that the user shall first select in the list of saved files.

The *Print Current* operation prints the current result (i.e the result of the last *Exec&Display* operation which occured in this window).

In both cases, the result is printed via **postscript.** The postscript files are in the directory *$TEV_HOME/data*. The user can use his own postscript files, which can be different for each result types. For this, install the customized postscript file in the directory *$VICOS_TEV_WD/RESULTS* for the data type which shall have its differentiated postscript file. Example : install a postscript file in the directory *$VICOS_TEV_WD/RESULTS/EVENT_LIST* for the events listings.

When no postscript files exist in the *$VICOS_TEV_WD/RESULTS/*data_type, TEV uses automatically the ones in *$TEV_HOME/data* :

- ***mp.pro.ln.ps*** (landscape mode) for the Data Set Listing and Merged Data Set / Events Listing
- ***mp.pro.pr.ps*** (portrait mode) for Events Listings, Statistics and Raw Data Dumps
- ***tev_graph_prolog.ps*** for the Graphs. This prolog file shall not be customized. Any postcript file in the directory *$VICOS_TEV_WD/RESULTS/GRAPHS* will be **ignored**.

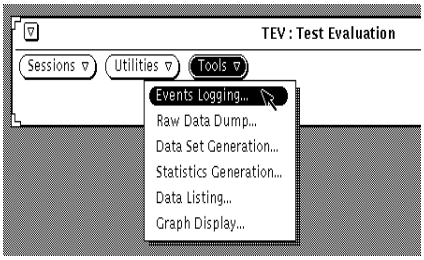
The *Exec&Display...* operation creates a result from the current definition. The definition is first checked for error. In case of error no execution takes place. During execution, further operations in the window are dis-

abled and this is shown by the shaded title–bar and stopwatch pointer. However, operations in other windows are permitted.
If the window has been unpinned and the definition has not been changed since the last execution, invoking the *Exec & Display...* operation will not generate the result again but simply display the previously generated result.
When the execution is complete, the result is displayed in a separate result window. The title of this window contains the tool reference and its instance number. The user can browse at the result in the usual OpenWindows way.

Exec & Display a definition

- Select **Definition –> Load...** from the pop–up menu.
- Click on the definition name in the list.
- Press the **Load** Button.
- Select **Result –> Exec & Display** from pop–up menu.

9.4.2 Events Logging Tool



To access *Events Logging* tool operations, the user must select the **Events Logging...** item in the ***Tools*** menu (Figure 9–17)

Figure 9–17 : *Start of the Events Logging Tool*

The *Events Logging* Tool window (Figure 9–18) appears:

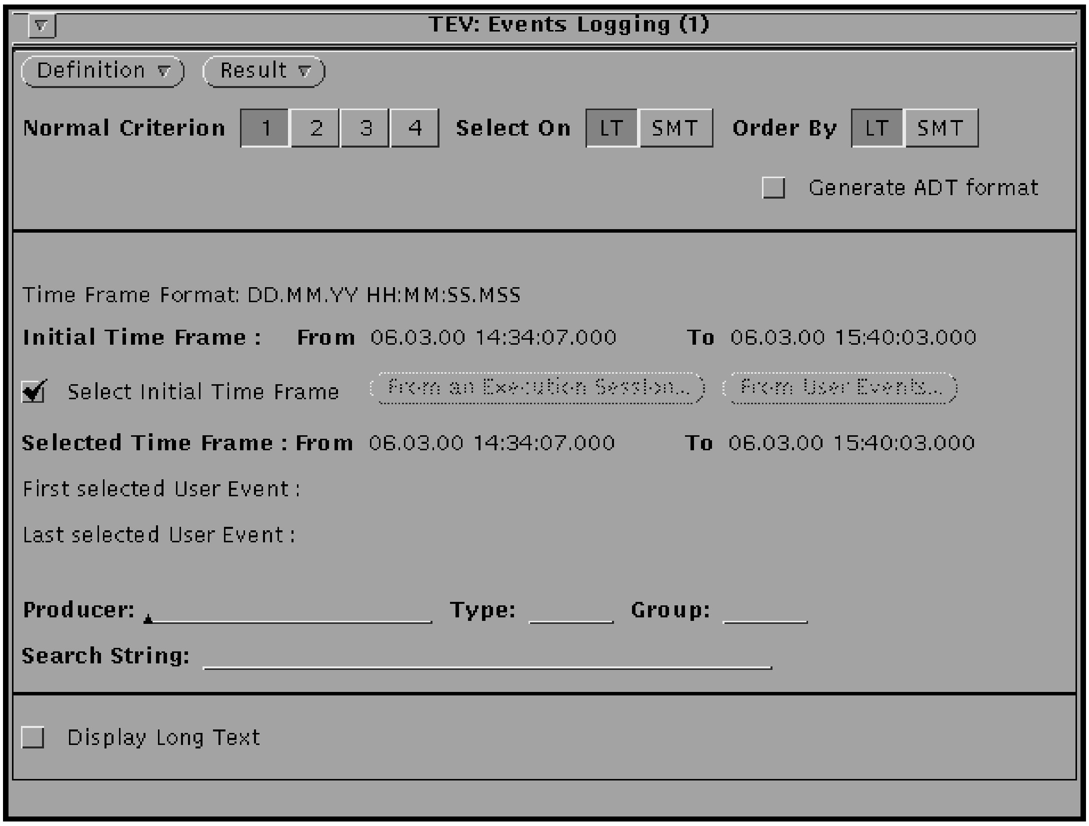


Figure 9–18 : *The Events Logging tool main window*

The **TEV: Events Logging** Tool main window shows the two menu buttons labelled **Definition** and **Result** and a panel with the contents of the current definition.

For more information about *Definition* and *Result* operations refer to the Chapter 9.4.1.

9.4.2.1 Create a definition

A Events Logging Tool definition allows the user to select events and to define the output format of events.

The user can either create a new definition or select and modify an already existing definition.

Note that it is not mandatory to give any selection criterion in the Events Logging Tool. The user can use the default definition without changing the entries in the ***Events Logging*** *main window. In this case* ***all*** *events which happened during the Initial Time Frame and for the selected sessions will be selected from the TRDB and put into the result.*

Levels of selection criteria :

There are three levels of criteria in the **Events Logging Tool**:

- An ***Atomic*** criterion is the simplest selection mechanism and allows to specify the value of a single field of a logged event. The ***Source*** field allows to include only events which originate from a given application. The ***Type*** and ***Group*** fields allow to extract only events of specific types (MSGE,UEVT...) and group (HCI,TES,...). The ***Search String*** field specifies a string which must be searched for in the text associated to events; only events containing the sub–string in their text field will be selected. The ***Selected Time Frame*** is also an atomic

criterion used to restrict the event selection in time.
If the text field of an ***Atomic*** criterion is left empty, all the events will match the criterion with regard to this field (i.e. It is the wildcard.)

- A ***Normal*** criterion is the combination of the different types of atomic criteria. The parts are joined by ”AND” relationships.
(e.g. (Time between t1 and t2) AND (Source = S1) AND (Type = T1) AND (Group unspecified) AND (Text includes ’abc’)).

Use of a specific Normal criterion

■ Select among the **Normal Criterion** boxes the number 1,2,3 or 4

The criteria fields appear empty, except for the time frame, set by default to the initial time frame. If these criteria fields are not modified, the normal criterion is considered as empty.

The Normal Criterion 1 is used by default, the others Normal criterion are by default empty.

- A ***Combined*** criterion is a combination of up to 4 *Normal* criterion. The Normal criterion are joined by ”OR” relationships. An empty *Normal* criterion is ignored. To be taken into account into the combination, a Normal criterion just has to be defined (not empty).
Example : if Normal Criterion 1,2 and 4 are not empty, events satisfying (Normal Criterion 1) OR (Normal Criterion 2) OR (Normal Criterion 4), are selected.

Select the Time Frame :

The **Select Initial Time Frame** check box is enabled by default. When this box is enabled the **Selected Time Frame** fields cannot be modified. To change the time frame the user must disable the check box first.

There are 3 alternatives to specify a time frame :

- Enter it directly in the *Selected Time Frame* fields,
- Copy it from the time frame of an Execution Session,
- Copy it from the time tags corresponding to 2 User Events.

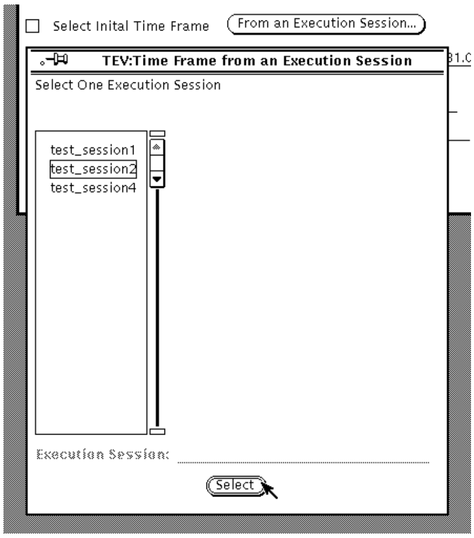


Figure 9–19 : *Selecting a Time Frame from an Execution session*

Enter directly a time frame.

- Disable the **Select Initial Time Frame** check box by clicking in the check box.

The Selected Time Frame *fields become available. The* Selected Time Frame: From *and* To *fields are shown underlined.*

- Enter a time frame into the **Selected Time Frame** fields.

The day (DD), month (MM), year (YY), hour (HH), minutes (MM) and seconds (SS) fields must be two–digit numbers. The milliseconds field (MSS) has three digits. Note that HH, MM, SS *and* MSS *have default value of 0 and are optional from the right to the left.*

Select a time frame from two User Events

- Disable the **Select Initial Time Frame** check box by clicking in the check box.
- Press the **From User Events...** button.

The window TEV: Time Frame from User Events *appears, containing the User Events existing in the currently selected Execution Sessions.*

- Select the Time Type using the **Time Type** boxes.
- Click on two user events.

The two time tags will be ordered and a time frame constructed.

- Press the **Select** button. The new time frame will be copied to the **Selected Time Frame** fields.

*The short texts of the two selected user events appear in the corresponding fields : **First Selected User Event** and **Last Selected User Event**.*

OR :

- Enter directly the User Events short texts in the corresponding fields : ***First Selected User Event*** *and* ***Last Selected User Event****.*

*The texts entered in the fields **First Selected User Event** and **Last Selected User Event** are part of the definition and will be saved when the definition is saved. This allows the user to work with different sessions, using the same definition of a Time Frame described through User Events . TEV uses the texts entered in the fields **First Selected User Event** and **Last Selected User Event** prior to the Selected Time Frame.*

To enable the Initial Time Frame *again, enable the* Select Initial Time Frame *check box.*
The Selected Time Frame *fields cannot be modified and hold a copy of the* Initial Time Frame

General procedure to specify the definition :

Specify an Events Logging definition

- Select the time base LT (local time) or SMT (simulated mission time) for **Select On** and **Order By** time types.
- Select a **Normal Criterion** 1,2,3 or 4
  - Select a Time Frame.
  - Type the logical name of the source in the **Source: ____** field.
    Only events which originate from a given source application will be selected.
  - Type the event type in the **Type ____** field.
    Only events of a given type (i.e. MSG) will be selected.
  - Type the group in the **Group ____** field.
    Only events which belong to a special group (i.e. user events) will be selected.
  - Type a string in the **Search String: ____** field.
    Only events which contain the given string will be selected.
- Define a set of criteria for each of the set 1,2,3 and 4
- Enable the **Display Long Text** check box. The long text of the events will be included in the result.

9.4.2.2 Generate a Result

By clicking on *Exec&Display* button in the *Result* menu, the tool will select the logging data from the Test Result Database according to the current definition.

The generation of the result (performed by the DBS tool) occurs in two steps : extraction of the events from the Oracle Database and formatting of the events in ASCII. The result is generated in a temporary file in the directory ***$TEV_HOME/data/tmp/*****tmp_machine_user**.
A window proposes the possibility to interrupt the ASCII formatting of the events. An interruption request can be taken into account only after completion of the Oracle Request.

- *Interrupt the Events Listing generation when the generation seems too long or when the disk space on $TEV_HOME is decreasing dramatically. In this case the user will get the result until the interruption.*

- *The Events Listing generation may be interrupted automatically because of a ”Disc Space problem”. In this case a result has been generated until the interruption. The user shall first recover space on the disc before saving the result by selecting* ***”Result:Save”.*** *Afterwards the user may recover some more place by deleting the temporary files in* ***$TEV_HOME/data/tmp/*****tmp_machine_user**. *The result can be then re–loaded normally by selecting* ***”Result:Load”*** *...*
  To avoid space problem when a big amount of events has been stored for the session, it may help first not selecting ***”Display long text”*** *and consider which time period is of interest.*
  Running TEV on the DBS–Server decreases considerably the processing time.

Events in the database that are included in the *Selected Time Frame* and originate from the given *Source*, of the given *Type (messages, alarms)*, of the given *Group (HCI, TES...)* and which contain the given *String*, will be included in the generated result. If a text field is empty, all the events will match the criterion with regard to this field.

The *Save...* operation copies the current result into the directory user_home*/wd/tev/RESULTS/EVENT_LIST.*

To save the result into the TRDB (i.e into an Evaluation Session) use the ***store*** *operation from the File Manager tool (see Chapter 9.5.1 File Handling).*

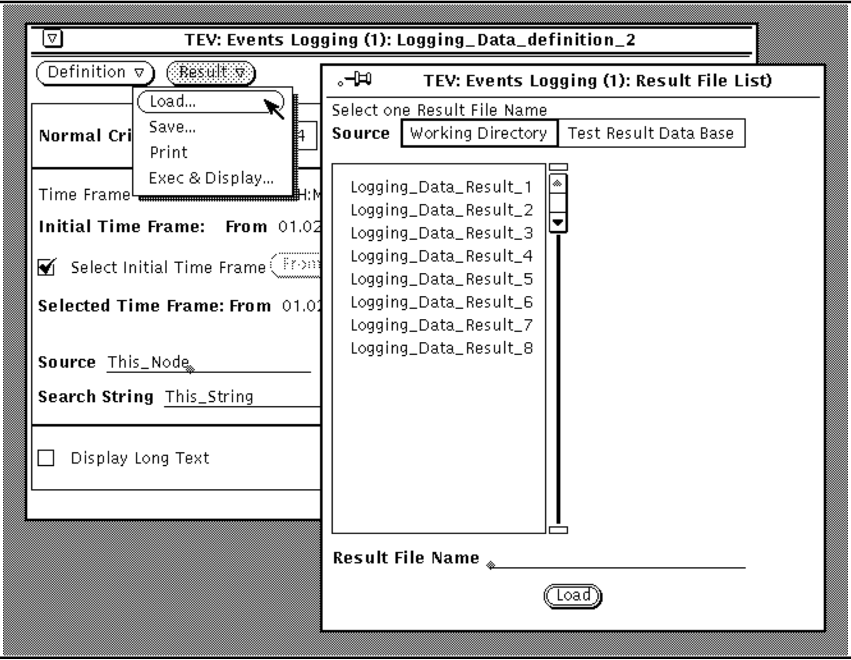


Figure 9–20 : *The result menu and the list of result files*

9.4.2.3 Generate an ADT Result

Instead of generating the result in an ASCII file the user can generate the result in an ADT format. The ADT format are ADA structures which are readeable througth ADT procedures. These ADT procedures are provided by the TEV_API library. Using this format allows the user to work with Events Listings within external ADA programs.

Generate an ADT Result

- specify a definition (see chapter 9.4.2.1)
- select the check box : **Generate ADT format**
- select **Result:Save...**
- Enter a filename for the ADT result and click on **Save** in the **TEV: Events Logging (i) : Result File Names** window

The result is directly stored in the directory $HOME/wd/tev/RESULTS/EVENT_LIST. The extension ***.adt*** *is automatically added to the filename. The user shall provide a filename of 16 characters maximum, because filenames are restricted to a length of 20 characters.*

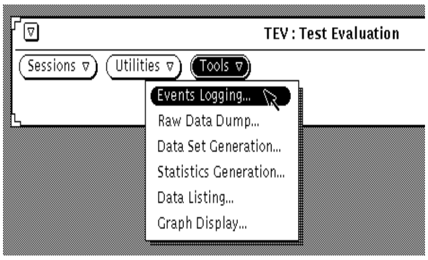
The ADT result can be displayed in ASCII in TEV by loading it : use ***Result:Load... ,*** *select the ADT filename and click on* ***Load****.*

9.4.3 Raw Data Dump Tool

Archive files (Raw data files) contains two data formats:

- incoming data are received and stored as ADU (**A**cquisition **D**ata **U**nits)
- outgoing data are stored as GDU (**G**eneration **D**ata **U**nits)

ADUs and GDUs are stored in the Archive files during the online operations and may be dumped by the Raw Data Tool after or during the test.



To access *Raw Data Dump* tool operations, the user must select the **Raw data Dump...** item in the ***Tools*** menu (Figure 9–21)

Figure 9–21 : *Start of the Raw Data Dump tool*

The **Raw Data Dump** window (Figure 9–22) appears:

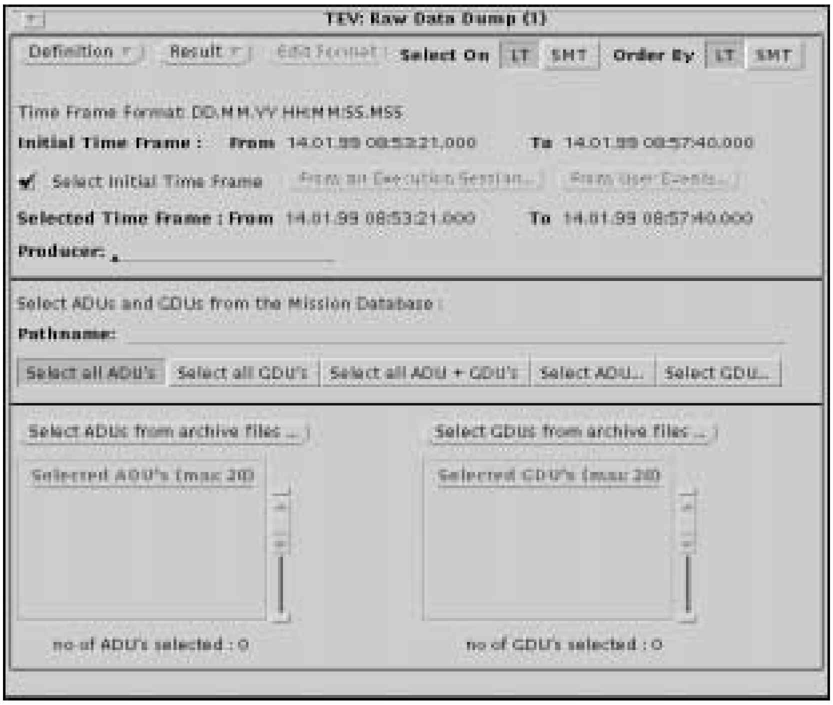


Figure 9–22 : *Raw Data Dump Main Window*

The Raw Data Dump tool **Exec&Display** functionality have been splitted in two different functionalities :

- ***Result:Start Packets Navigator*** : to consult directly the packets dumped in the Archive Files (No result files are generated)
- ***Result: Save Dump in file*** : extract packets and save them in a result file (in ASCII or ADT format)

*The result files are now per default in ASCII. To re–use (**Result:Load...**) the old ADT result files, rename them first with the extension .adt.*

In the previous version TEV was first generating a file containing the extracted items and the user could then consult the items dumped in this file. The time consumed to generate this file and the size of the generated file was proportional to the size of the Archive Files. Now TEV searchs the packets to display directly in the Archive Files.

Use first the Packets Navigator (or up to 4 Packets Navigators) to analyse the relevant data. Reduce the Time Frame before starting the Packets Navigator(s) in order to optimize your investigation.

Use than the option ***Result: Save Dump in File*** to generate a result file containing the dumped packets of interest.

TRDB or Foreign Database
At initialization time, TEV read a configuration file ($TEV_HOME/config/tev_configuration.data) to determine from which database will be read the Archive Files : TRDB or a Foreign Database. By default the database is TRDB. The Database in used is set–up once for a whole TEV session and for all the TEV users on the environment. (The same configuration file is used across all the TEV processes).

9.4.3.1 Create a Definition

Create a Definition

- Select a Time Frame to limit the data to be dumped or displayed.

The selection of a Time Frame is identical in each Evaluation Tool. Refer to "Select The Time Frame" paragraph in the Chapter 9.4.2.1 Events Logging Tool.

- If working with the Foreign Database, **Select On** and **Order By** SMT time tags.
- Type the selected producer, i.e. the name of the test node having archived the raw data to extract, in the **Producer : ___** text field.
- Select the data to be dumped or displayed. For this two possibilities :
- EITHER
- Select the ADUs/ GDUs from the Mission Database (Figure 9–23). This choice makes sense when the user knows exactly the pathname of the items to be selected, or when the user only wants to create a definition and has selected no session (and consequently no archive files).
- Fill the **Pathname:____** text field with a virtual node name, which will be the root for the ADUs and GDUs parameters to be selected. The user can get quickly the pathname of the dumped items in the Archive files by doing the following :
  - Click on ***Edit Format*** and choose ***Summary Format.*** Quit the menu by clicking on ***Accept.***
  - Start ***Result : Start Packets Navigator*** and eventually interrupt the dump : The list of the dumped items (with time tags and without contents) appears. From this list the user can extract the relevant pathname(s).

*The "All ADUs", "All GDUs" and "All ADUs and GDUs "options will select all the wanted parameters existing under the pathname selected. These three buttons are mutually **exclusive**.*

The "Selected ADUs" and the "Selected GDUs" options will open a window to choose the parameters among a list of all parameters existing under the pathname selected.These buttons can not be combined with one from the other list : "All ADUs", "All GDUs" or "All ADUs and GDUs". For example, it is not possible to choose "All GDUs" and "Selected ADUs". In this case, the user shall choose "Selected GDUs" and "Selected ADUs" and select himself all the GDUs.

OR

- Select the ADUs/ GDUs from the Archive files. (Figure 9–24).
- Press the **Select ADUs from Archive Files...** button.

A message warns the user if the Archive files global size is over a certain limit.

Create a Definition (continue)

- *TEV consults all the archive files for the selected sessions and the selected time frame.This action can take a long time.That is why the message warns the user, if the global size of the archive files reached a certain size. This size is configurable in the file **$TEV_HOME/config/tev_configuration.data** through the parameter **LIMIT_ARCHIVE_FILES_SIZE.** The time consumed for this action : **Select ADUs from Archive files** is machine dependent : the user shall find out up to which size it is reasonable to consult the Archive files in his environment.*
- Proceed searching the ADUs
- OR
- Abort the action
- *In case of aborting the action, the user is supposed to either select the ADUs from the Mission Database or reduce the time frame. Becarefull ! : When reducing the time frame to a short interval concerning only a part of one archive file, TEV will nevertherless consider the global size of the archive file for warning the user. In this case it can make sense to ignore the warning message.*
- If proceeding, the list of the ADUs archived appears. Select in this list the ADUs to be dumped.
- Press the **Select GDUs from Archive Files...** button. The list of all the GDUs archived appears. Select in this list the GDUs to be dumped.
- A message warns the user if the Archive files global size is over a certain limit.
- *Same behaviour as with **Select ADUs from Archive files***
- If proceeding, the list of the GDUs archived appears. Select in this list the GDUs to be dumped.
- Choose the output format for the packets (see procedure **Select the output format for the packets** in Chapter **9.4.3.2**)

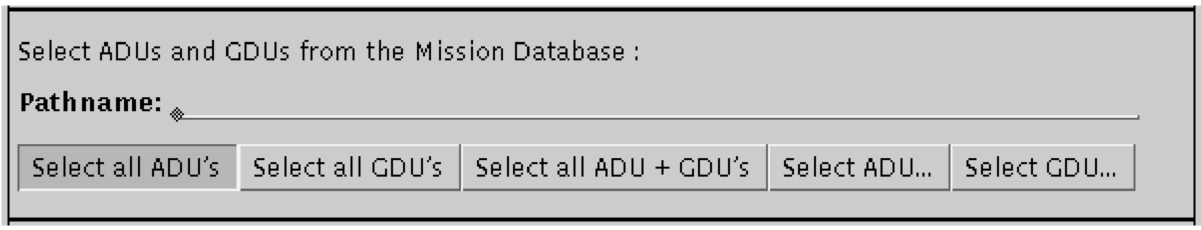


Figure 9–23 : *Select the ADUs and GDUs from the Mission Database*

Figure 9–24 : *Select the ADUs and GDUs from the archive files*

9.4.3.2 Consult the packets dumped in the Archive Files

Consult the packets dumped in the Archive Files

- Create a Definition. The definition can be empty, in this case all the ADUs and GDUs are selected.
- Select **Result : Start Packets Navigator** from the pop–up menu.

The navigation tool appears. No result file is generated : the Packets Navigator is foreseen ***only to consult*** *the packets in the Archive Files. The time consumed to open the Packets Navigator is only the one needed to find the first packet corresponding to the selected item list. For example, if the user has selected all the ADUs and GDUs from the Archive files, the Packets Navigator appears immediately.*

- Navigate through the packets with the navigation tool :

The user can inspect the packets only one by one. To compare some packets, open a second Raw Data Dump tool and proceed like described to start a second Packets Navigator.

Press the **Next/Previous** button to display the next/previous packet in the Data dump

OR

Type a time tag in the text field beneath the **Jump To** button.

Press the **Jump To** button to display the first packet for which the Archiving Time Tag is equal or bigger to that time.

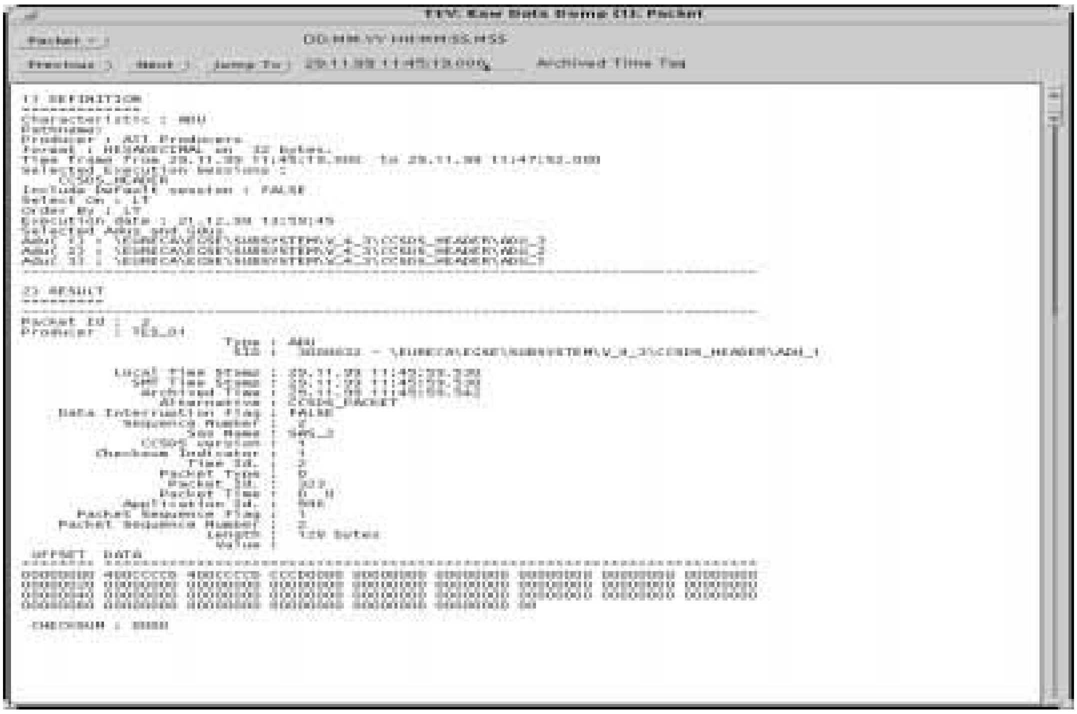


Figure 9–25 : *A Raw Data Packet and the 'navigation' buttons*

The lower part of (Figure 9–25) is a scrollable text zone with the contents of the current packet. This sub–window is read–only.

The format of the packets may be changed when the Packets Navigator is already open. Use the button **Packet:Edit Format ...** option from the Packets Navigator window or the **Edit Format** button from the main window. The *Output Format* pop–up window (Figure 9–26) appears :

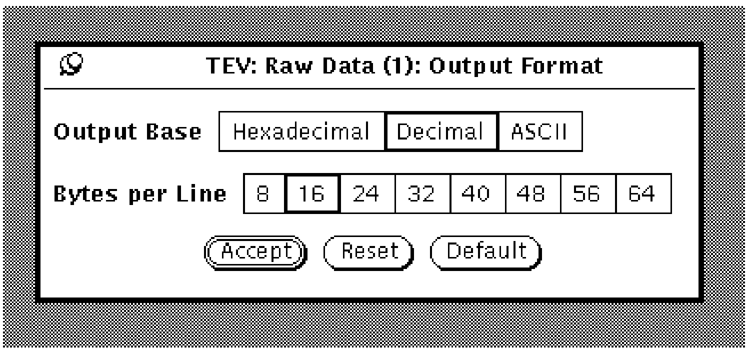


Figure 9–26 : *Output format of a Raw Data dump*

Select the output format for the packets

- Press **Edit Format** button.
- Select the output base: Hexadecimal, Decimal or ASCII
- Select the number of bytes per line
- Press the **Accept** button.

To go back to the default format (Hexadecimal, 32 bytes per line) click on the ***Default*** *operation.*

The format specified is used for packets and complete dumps, when the data is a byte stream (i.e. for unstructured data, CCSDS packets).

The layout of the data dump may also be specified within the Packets Navigator :

Reformat a Raw Data packet

- In the navigation tool press the **Packet** button and select **Packet : Edit Format...** from pop–up menu.
- Change the output format in the Output Format window (Figure 9–26)
- Press the **Accept** button in the Output Format window.

9.4.3.3 Consult the packets in the Archive Files in a summary format

The summary format is obtained through the **Edit Format** button of the Raw Data Dump main window. It allows the user to get an overall view of the packets dumped in the archive files. A summary format can not be generated when the Packets Navigator is already open. (Close the Packets Navigator).

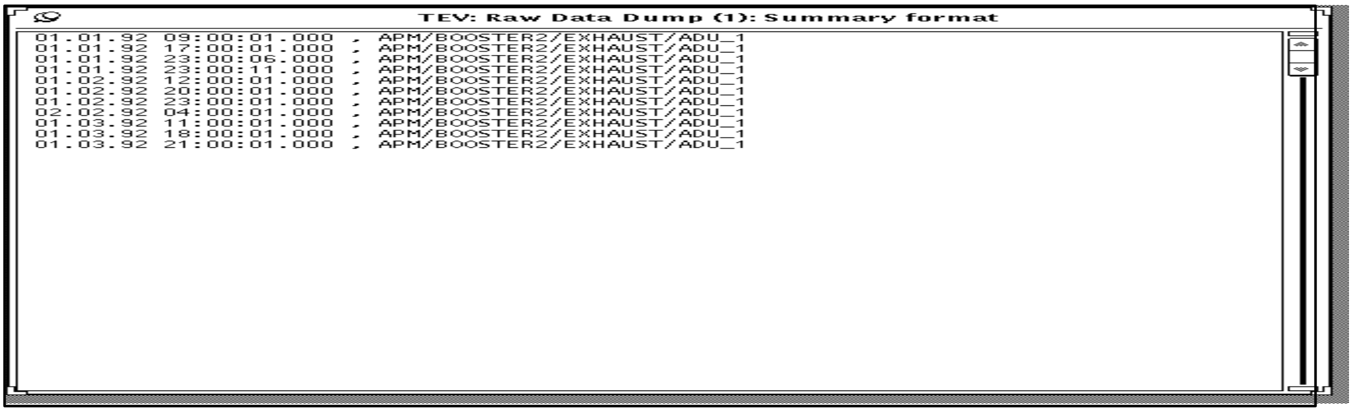


Figure 9–27 : *A Summary Format*

Consult the packets dumped in the Archive Files in a summary format

Create a Definition

Click on **Edit Format** and choose in the window **Output Format** the option : **Summary Format**.

Click on **Accept** in the window **Output Format**

Select **Result : Start Packets Navigator** from the pop–up menu.

A window pops–up and proposes an Interrupt button. TEV reads all the Archive Files, generates and displays the result. No result file is generated : use ***Result : Save Dump in File*** *with the option* ***Summary Format*** *to generate a summary result file. Interrupt the dump when the generation seems too long or when the disk space on $TEV_HOME is decreasing dramatically. The result is generated until the interruption.*

The Raw Data Dump generation may be interrupted automatically because of a ”Disc Space problem”. In this case the result is generated until the interruption.

9.4.3.4 Save the selected packets in a result file

Save a Dump in a result file

Create a Definition. Different options can be used in the definition :

- **Edit Format :** check box **Summary Format** selected

OR

- Check Box **Generate ADT format** selected

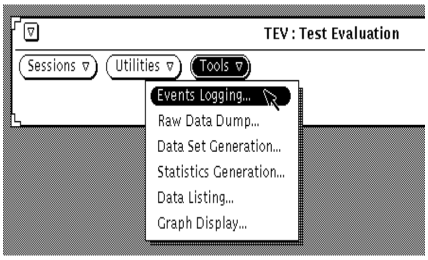
No summary format in an ADT format

Save the result : **Result:Save Dump in File** ... The result file is generated in the directory **$HOME/wd/tev/RESULTS/RAW_DATA_DUMP** (with the extension “.adt” when the ADT format has been selected).

A windows pops–up and proposes an Interrupt button.Interrupt the dump when the generation seems too long or when the disk space on $TEV_HOME is decreasing dramatically. The result is generated until the interruption.

The Raw Data Dump generation may be interrupted automatically because of a ”Disc Space problem”. In this case the result is generated until the interruption. The result can be afterwards displayed normally by selecting ***”Result:Load”*** *... (Recover space before loading an ADT result file, because this file will be first formatted in ASCII).*

9.4.4 Data Set Generation Tool



To access *Data Set generation* tool operations, the user must select the **Data Set generation...** item in the ***Tools*** menu (Figure 9–28)

Figure 9–28 : *The **Tools** menu*

This tool allows the user to create data sets containing raw value and their associated calibrated engineering value. The values are extracted from either the archived files or from the engineering value logbooks.

TRDB or Foreign Database
At initialization time, TEV read a configuration file (*$TEV_HOME/config/tev_configuration.data*) to determine from which database will be read the Archive Files : TRDB or a Foreign Database. By default the database is TRDB. Furthermore, in case the Foreign Database is used, two options can be set–up through this configuration file :

- TEV generates the Data Sets reading the Archives Files provided by the Foreign Database : option INTERNAL generation
- The Foreign Database provides the Datasets : option EXTERNAL generation

The configuration in used is set–up once for a whole TEV session and for all the TEV users on the environment. (The same configuration file is used across all the TEV processes).

The *Data Set Generation* main window is presented in Figure 9–29 :

*Note that there is no **Result** menu as a Data Set is not directly displayed or printed in this tool. Data Listing, Statistics and Graph tools are provided for this task.*

For more information about *Definition* operations refer to Chapter 9.4.1.
The upper part of the *Data Set Generation Window* (Figure 9–29) contains the Time Frame selection area, described into chapter *9.4.2.1 Events Logging Tool, Select Time Frame* paragraph.

Figure 9–29 : *Data Set Generation Main Window*

TEV allows to generate a data set, i.e. a set of values for selected measurements (maximum 50) within a specified time frame. Values may be sampled to take all from the time frame or only every nth one. The result is written to a Data Set file.

9.4.4.1 Select the sampling mode

Select the Every n Sampling Mode

- Press the **Sampling** menu button.
- Select **Sampling –> Every n** from the pop–up menu.
- Type the number for selecting every nth sample in the **Every: ____** text field.

*If the user wishes to include all data the entry in the **Every**: field is 1.*

When ”n sampling” has been defined, TEV insert in the Data Set each nth value found for each parameter.

Select the Time based Sampling Mode

- Press the **Sampling** menu button.
- Select **Sampling –> Time Based** from the pop–up menu.
- Set up the **Time Interval** and **Allowed Error fields.**

Figure 9–30 shows the sample selection in the Time Based mode.

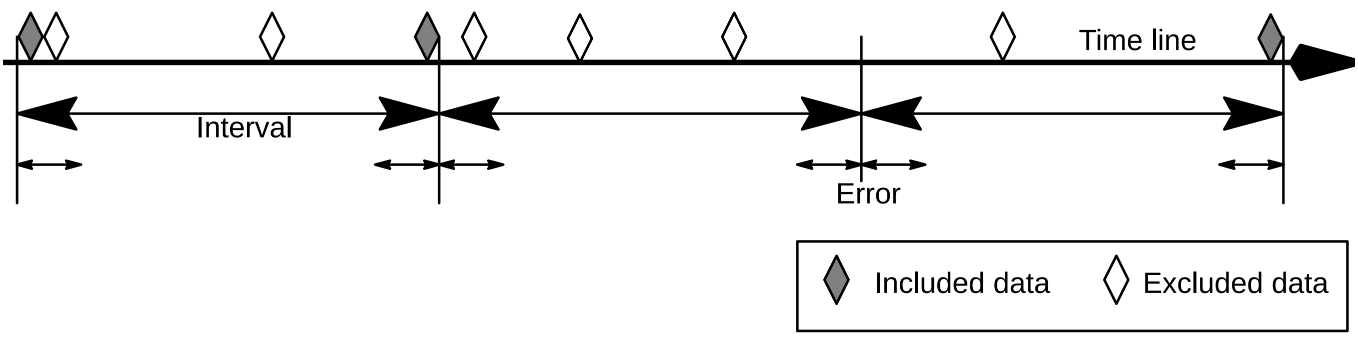


Figure 9–30 : *Time based sampling semantics*

When a time based sampling has been defined in the global time frame [T0 .. T1] as : the time interval is DeltaT and the error is Eps, TEV extract the value for each parameter which is in the intervals :
– [T0 .. T0 + Eps/2] for the first value
– [T0 + mDeltaT – Eps/2 .. min(T1, T0 + mDeltaT + Eps/2)] for the last value (m is the greatest number where T0 + mDeltaT <= T1)
– [T0 + iDeltaT – Eps/2 .. T0 + iDeltaT + Eps/2] for the intermediate values (i = 1 .. m–1)

The Allowed Error interval assures that data which have a slight deviation at the time of their occurrence are included in the sample.A value is selected if it lies within the error limits.
If there are more than one values within the allowed interval, TEV extract the nearest value to the time tag (T0 + iDeltaT).

In the current implementation following problem arises:
If the sampling rate is set to 0 (zero) the data set is build without error message, but the data set is empty.

9.4.4.2 Select Data Set parameters

The parameters selected are the measurements from which the values have to be included into the Data Set. Up to 4 sets of parameters can be specified. However, only one set is displayed at a time. The parameter set to display is selected by the ***Parameter Set*** choice box. The user shall choose the numbers 1 to 4 to identify the current parameters set to display.

Define a parameters set

- Select from the **Parameter Set** check boxes the identification number of the set : 1,2,3 or 4.
- Select the source type : **Archive** or **Engineering Value Logbook**

The Engineering Value Logbook contains those values which were already calibrated during the test, the archive files contain raw data. When working with the Foreign Database, the source type shall be ***Archive****.*

- Specify the test node that produced the data to be extracted in the **Producer: ____** text field.
- Enter the node name in the **Virtual Node Name: ____** text field.

The Virtual Node Name defines the root pathname (Sub–Tree) for the measurements to be selected.

- Click on the **Select from Sub–Tree...** button.
- Wait until the **TEV: Data Set**(i) :**Parameter Names** subwindow pops up.
- Select parameter names from the list (Figure 9–31).
- Press the **Add** button.

The selected measurements appear in the tool main window, the subwindow **TEV: Data Set**(i) :**Parameter Names** disappears.*The double entries are automatically ignored.*

OR

- Click on the **Select All from Sub–Tree** button.

All the measurements under the Virtual Node Name will be selected. The double entries are automatically ignored.

OR

- Enter a parameter name into the **Parameter** text field.
- Click on the **Add** button.

The measurement is added into the list, if it is no already in the list.

Click on the **Check Parameter Set** button : TEV will automatically check the existence and validity of the parameters selected in the currently displayed parameters set in the selected CCU. If some parameters are invalid the user shall remove them from the list.

In case a measurement is acquired by multiple ADUs, it is possible to select which ADUs shall be consider for the measurement :

Select a measurement in the list of selected measurements.

Click on the **Choose ADUs from Archive files...** button. The list of ADUs appearing in the archive files and which acquire the selected measurement appears. Click on the wanted ADUs in the list to select them.

OR

Define a parameters set (continue)

- Click on the **Choose ADUs from MDB..** button. The list of all ADUs from the MDB which acquire the selected measurement appears. Click on the wanted ADUs in the list to select them.

- *The name of an already selected parameter can be modified. Select the parameter from the list : the name is copied into the* ***Parameter*** *field. Modify the parameter name and click on the* ***Replace*** *button (which is enabled only when a parameter is selected).*

- *Parameters may be removed from the list by selecting them and clicking on the* ***Remove*** *button.*

Note that selection into the list of selected measurements in the main window has no meaning for data set generation or for check parameter set.

A *Total Parameters Selected* count field displays the **total** number of selected parameters in the four parameter sets.

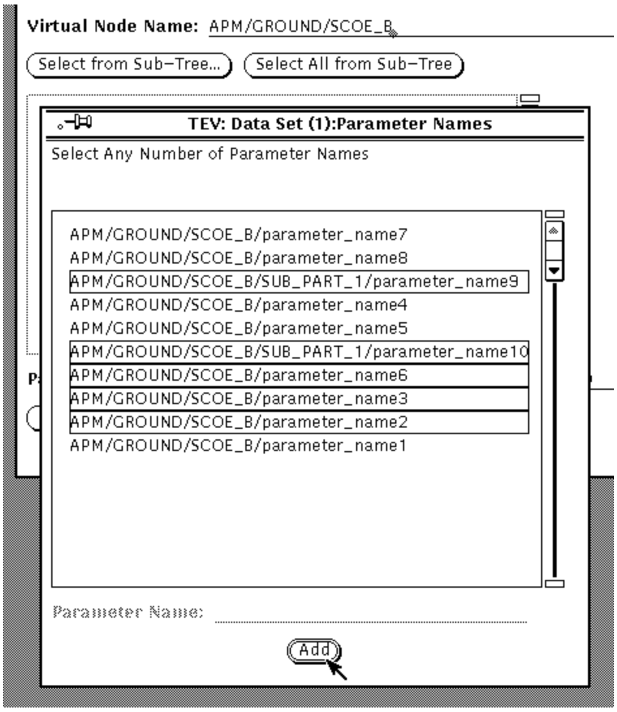


Figure 9–31 : *Selection of parameters below the Virtual Node Name tree*

Generate the Data Set

- **Select On : SMT** and **Order By : SMT** if working with the Foreign Database.
- Click on the **Build Data Set ...** button.
- Enter the new data set name.

A data set name is made up of a maximum of 20 alphanumeric characters

- Click on the **Build** button
- A windows pops–up and proposes an Interrupt button

*Interrupt the data set when the generation seems too long or when the disk space on $TEV_HOME is decreasing dramatically. In this case the user will get **no result**.*

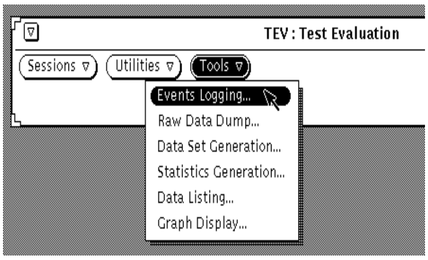
The Data Set is generated for the combination of the measurements selected in the four parameters sets. When the generation is completed, the Data Set is available for statistic generation, data listing and graph display.

*When some problems occured, TEV displays a text–editor containing a report about the generation. The file containing the report is in the directory **$TEV_HOME/data/tmp/tmp_**__**machine_user**__.*

The data sets merger tool allows two data sets that were built with the same time type criteria to be merged, so that data from differnent test sessions shall be mixed, ordered by time tags.

To access the Data Set Merger, the user must select the ’**Data Set Merger...**’ item in the *Utilities* menu. (See Chapter 9.5.2).

9.4.5 Statistics Generation Tool



To access *Statistics Generation* tool operations, the user must select **Statistics generation...** item in the ***Tools*** menu (Figure 9–32)

Figure 9–32 : *The **Tools** menu*

The results generated here do not depend on the selected Execution Sessions, but on a selected Data set. Before the user can perform a Statistic Generation the actions described in Chapter 9.4.4 must be performed, e.g. a Data Set has to be created.

Statistics Generation tool (like the Data Listing tool and the Graph Display tool) works on data sets only.

The *Statistics Generation* main window is presented in Figure 9–33 :

For more information about *Definition* and *Result* operations refer to the Chapter 9.4.1.

Note that when the Tool is started a data set called *default_data_set* is loaded from the user's working directory as default selected data set and all its parameters will be displayed in the panel. The user can redefine the *default_data_set* by executing a Data Set Generation Definition and build a data set called *default_data_set*, overwritting the previous one. A *default_data_set* has to exist into the user_home/*wd/tev/RESULTS/DATA_SET* directory, otherwise an error message is displayed. This error message can be simply acknowledge without further consequences.

The *Initial Time Frame* is replaced by the *Overall Time Frame* which is the overall time frame of the Data Set from which data were extracted.

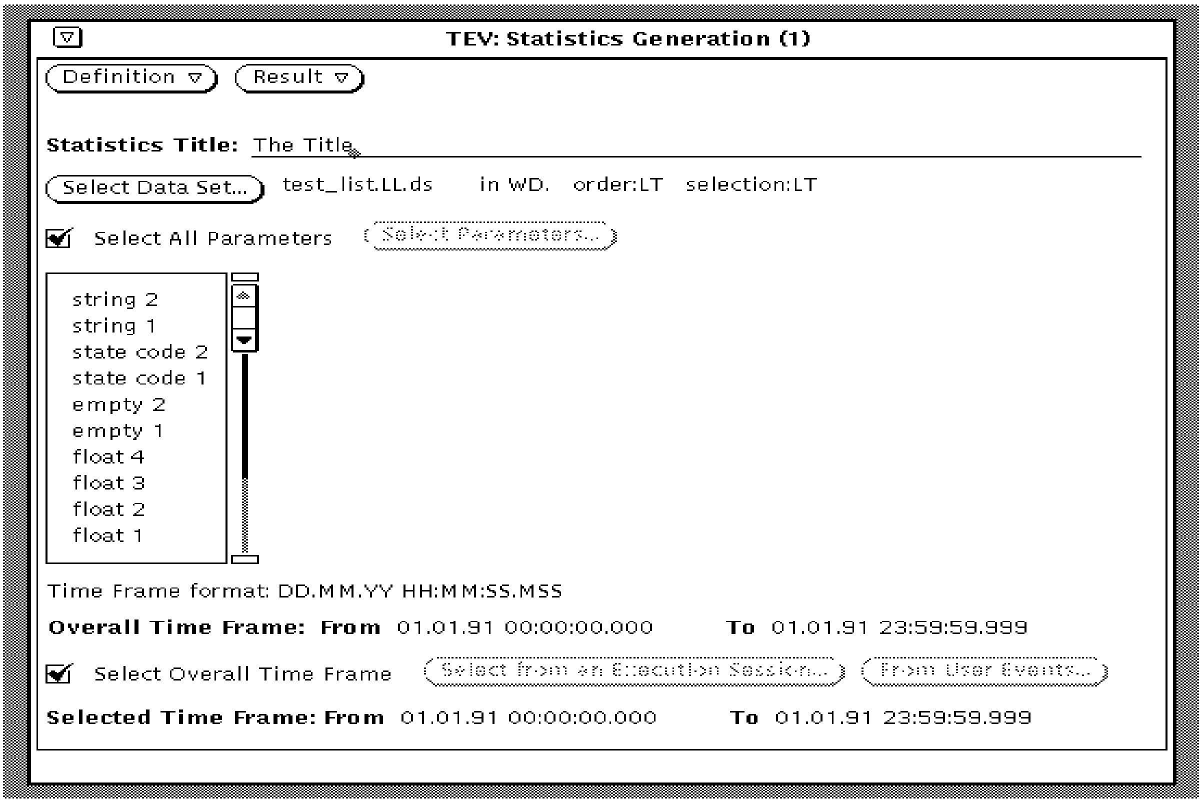


Figure 9–33 : *The Statistics Generation Tool window*

Build a Statistics Generation definition

- Type the name of the statistic in the **Statistics Title: ____** text field. This name will be included in the result file.
- Click on the **Select Data Set...** button.
- Select the desired data set.
- Press the **Select** button.
- Disable the **Select All Parameters** check box.
- *By default the* ***Select All Parameters*** *check box is enabled :* all the measurements in the data set are then selected (A data set may have up to 50 measurements).
- Press the **Select Parameters...** button.
- *The window* ***Statistics Generation (i) : Data Set Parameters*** *appears, containing the list of all the measurements included into the selected Data Set.*
- Select the desired parameters from the list.
- Press **Select** button. The reduced list is displayed in the **Statistics Generation** window.
- Reduce the time frame to limit the data included in the statistics : disable the **Select Overall Time Frame** box and select a time frame either form an execution session, either from User Events.

- *Once the wanted Statistics Generation Definition is defined, click on Result–> Exec&Display to produce the result.*
- *Note that for a given set of parameters a statistic is generated which contain maximum and minimum values, mean, median, variance and the number of measurements.*

9.4.6 Data Listing Tool

To access *Data Listing* tool operations, the user must select the **Data Listing...** item in the *Tools* menu (Figure 9–32).

The results generated here do not depend on the selected Execution Sessions, but on a selected Data set. Before the user can perform a Data listing the actions described in Chapter 9.4.4 must be performed, e.g. a Data Set has to be created.

- *The Data Listing tool (like the Statistics Generation tool and the Graph Display tool) works on data sets only.*

For more information about *Definition* and *Result* operations refer to the Chapter 9.4.1.

Note that when the Tool is started a data set called *default_data_set* is loaded from the user's working directory as the default selected data set and all its parameters will be displayed in the panel. The user can redefine the *default_data_set* by executing a Data Set Generation Definition and build a data set called *default_data_set*, overwritting the previous one. A *default_data_set* has to exist into the user_home/*wd/tev/RESULTS/DATA_SET* directory, otherwise an error message is displayed. This error message can be simply acknowledge without further consequences.

The *Initial Time Frame* is replaced by the *Overall Time Frame* which is the overall time frame of the Data Set from which Data are extracted.

The Data Listing tool is very similar in use to the Statistics Generation tool, the structure of the generated result only differs. The procedure to select measurements from the Data Set is the same. Up to 10 parameters can be selected. ASCII parameters values are truncated, except if the ASCII parameter is the ONLY ONE parameter selected. In this case, the ASCII parameter is displayed completely. If the check–box **Select All Parameters** is enabled, the first parameter(s) (same limits) are listed and a warning is logged.

9.4.7 Graphic Tool

A graph is generated from a selected Data Set. A graph result is a postscript file which may be displayed through TEV.

To access *Graph* tool operations, the user must select the **Graph Display...** item in the ***Tools*** menu (Figure 9–32).

For more information about *Definition* and *Result* operations refer to the Chapter 9.4.1.

Note that when the Tool is started a data set called *default_data_set* is loaded from the user's working directory as the default selected data set and all its parameters will be displayed in the panel. The user can redefine the *default_data_set* by executing a Data Set Generation Definition and build a data set called *default_data_set*, overwritting the previous one. A *default_data_set* has to exist into the user_home/*wd/tev/RESULTS/DATA_SET* directory, otherwise an error message is displayed. This error message can be simply acknowledge without further consequences.

The *Initial Time Frame* is replaced by the *Overall Time Frame* which is the overall time frame of the Data Set from which Data are extracted.

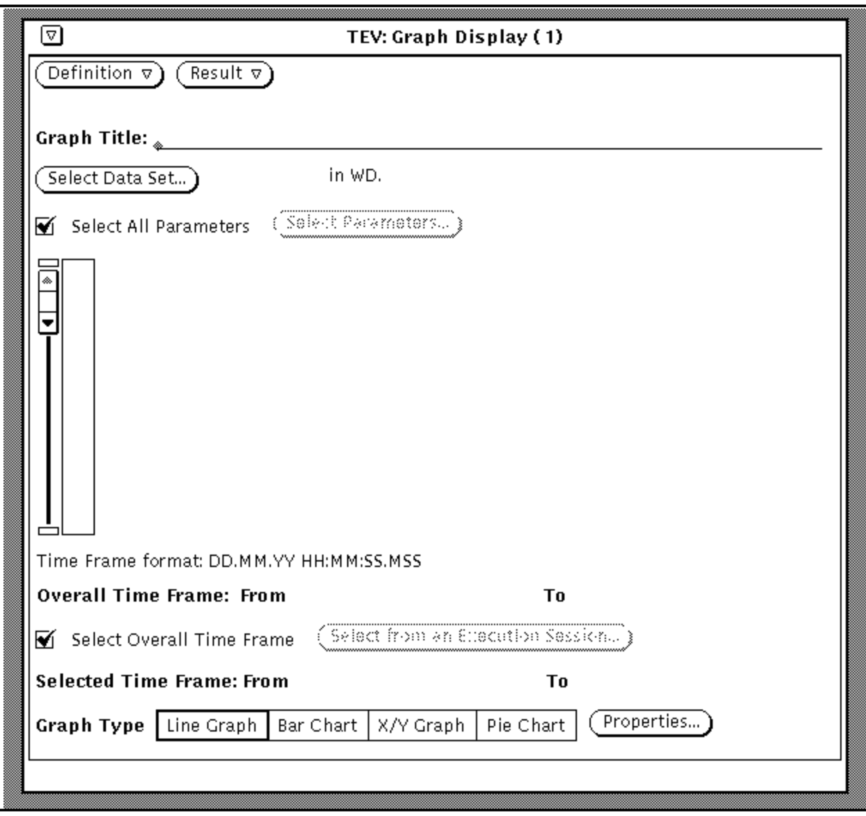


Figure 9–34 : *The Graphs tool window*

At the bottom of the window the **Graph Type** choice boxes allow the user to select the type of graph for the current definition.

Build a Graph Display definition

- Type the name of the graph in the **Graph Title: ____** text field. This name will be included in the result file.
- Click on the **Select Data Set...** button.
- Select the desired data set.
- Press the **Select** button.
- Disable the **Select All Parameters** check box.
- *By default the **Select All Parameters** check box is enabled : once the data set has been selected the first 5 parameters from that data set will be displayed in the parameters list. If the data set contains more than 5 parameters a message to indicate that the list has been truncated will appear in the footer of the graph display window.*
- Press the **Select Parameters...** button.
- *The window **Graph Display (i) : Data Set Parameters** appears, containing the list of all the measurements included into the selected Data Set (see Figure 9–35).*
- Select the desired parameters from the list.
- *The graph tool prevents more than 5 parameters being selected from this list.*
- Press **Select** button. The list of selected parameters is displayed in the **Statistics Generation** window.
- *Selection in this list has no meaning.*
- Reduce the time frame to limit the data included in the statistics : disable the **Select Overall Time Frame** box and select a time frame either from an execution session, either from User Events.
- Select the **Graph Type** to be displayed.
- Click on the **Properties** button to define the graphical properties for the type of graph chosen.

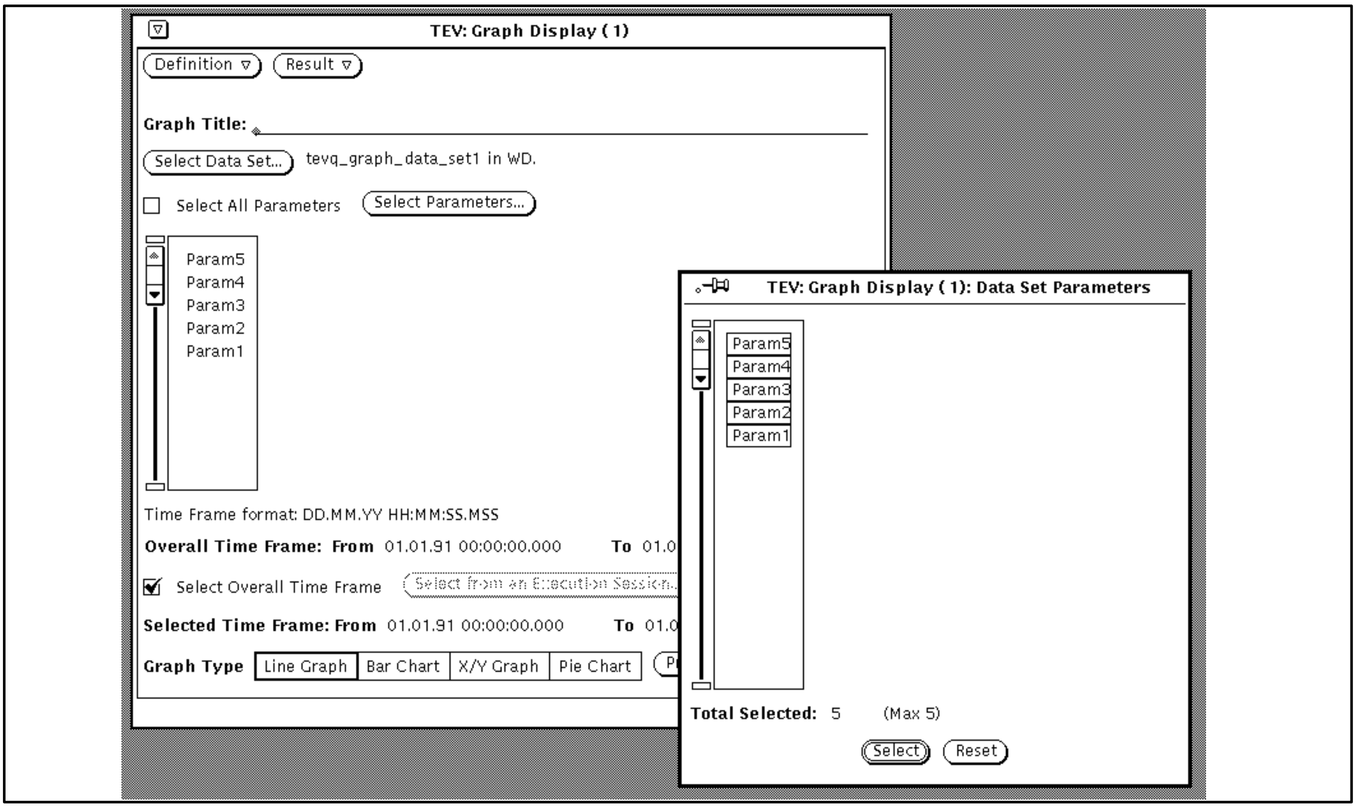


Figure 9–35 : *The pop–up window with the data set parameters*

There are four types of graph which the user may select :

- **Line Graph**. The Line Graph is a graph of up to 5 parameters against either absolute time or relative time.
- **Bar Chart**. The Bar Chart is calculated for the first value found in the time frame specified for each parameter selected.
- **XY Graph**. The XY Graph is a graph plotted of parameter x against parameter y.

If an **XY graph** is requested and more than two parameters appear in the selected list then the first two parameters of the list will be plotted. If only on parameter is selected then it will be plotted with the second parameter of the data set.

- **Pie Chart**. The pie chart is calculated for the first value found in the time frame specified for each parameter selected.

For a **pie chart** at least two parameters must be defined. The first value to be found within the time period defined will be charted for each parameter. The values must be of the same sign and type or an error will be reported.

Edit the Line Graph properties

- Press the **Properties...** button.
- Type the X Axis lettering in the **X Axis name:___** text field (max. 20 characters) (see Figure 9–36).
- Enable the **Time Frame** selection check box to **RELATIVE** or **ABSOLUTE.**
- *If RELATIVE, then the user can enter for each parameter a relative time frame.*
- Type the Y Axis lettering in the **Y Axis name:___** text field (max. 20 characters).
- Enable the **Scaling** selection check box to AUTOMATIC or MANUAL.
- If Scaling mode is MANUAL :
- Type the lower limit value in the **From:___** text field.
- Type the upper limit value in the **To:___** text field.
- *Values outside the range selected will not be plotted.*

The window *Line Graph properties* presents five identical paragraphs, each paragraph (i) allow to define the properties for the parameter (i) :

- For each parameter:
- Press the **Colour** button with the right mouse button and select a colour from the pop–up menu.(see Figure 9–37)
- Press the **Line Style** button with the right mouse button and select a style from the pop–up menu.
- Press the **Thickness** button with the right mouse button and select the line thickness from the pop–up menu.
- If the RELATIVE Time Frame selection is enabled :
- Type the selection start time in the **Relative Time Frame From:___** text field.
- Type the selection stop time in the **Relative Time Frame To:___** text field.
- *The to and from fields control the data selection. TEV displays the first value selected for each curve at reference start time frame and displays subsequent values relative to this up to the end time.*

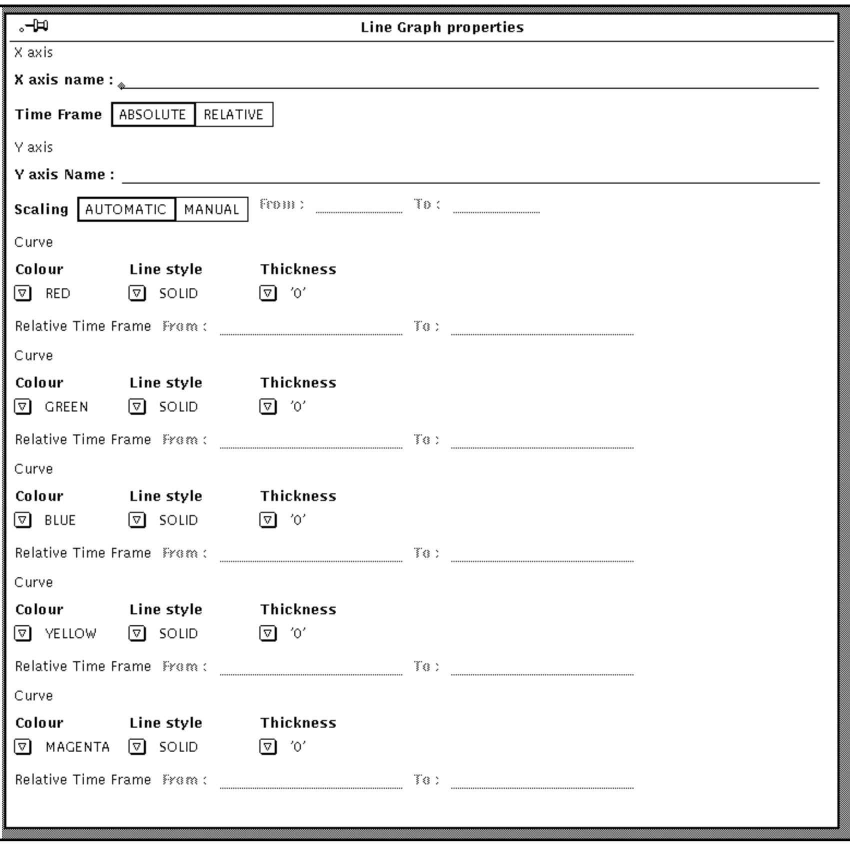


Figure 9–36 : *Line Graph properties window*

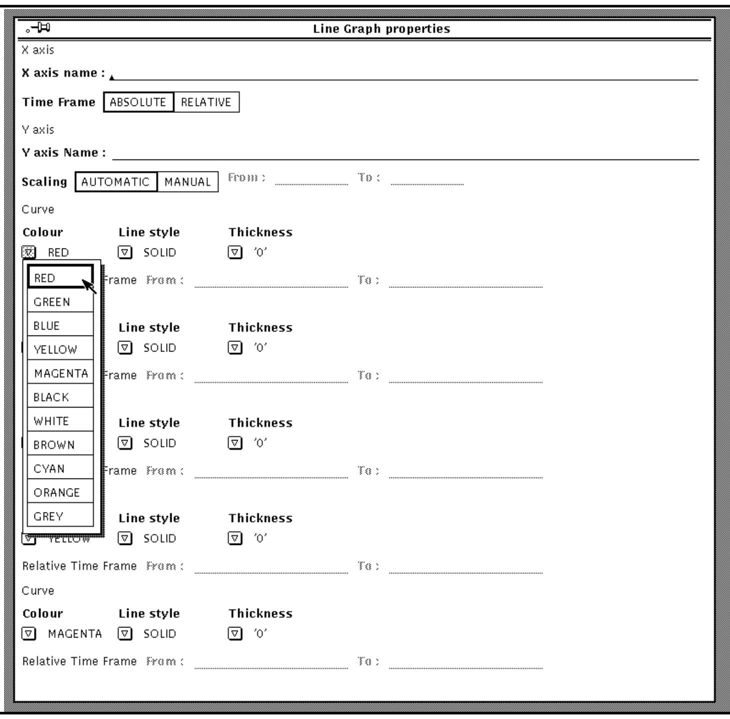


Figure 9–37 : *Line Graph properties window with colour selection*

The **Bar Chart** Properties window comprises only a colour menu which allows to select up to 11 different colours for each parameter.

Edit the XY Graph properties (see Figure 9–38)

- Click on the Graph Type selection button **XY Graph**.
- Press the **Properties...** button.
- Type the X Axis lettering in the **X Axis name:___** text field.(max. 20 characters)
- Type the Y Axis lettering in the **Y Axis name:___** text field. (max. 20 characters)
- Enable the **Manual Scaling** check box for the X axis.
- Type the lower range in the **From:___** text field.
- Type the upper range value in the **To:___** text field.

Values inside the range selected will be indicated plotted.

- Enable the **Manual Scaling** check box for the Y axis.
- Type the lower range in the **From:___** text field.
- Type the upper range value in the **To:___** text field.

Values inside the range selected will be plotted.

- Click on the **Colour** button and select a colour from the pop–up menu.
- Click on the **Line Style** button and select a style from the pop–up menu.
- Click on the **Thickness** button and select the line thickness from the pop–up menu.

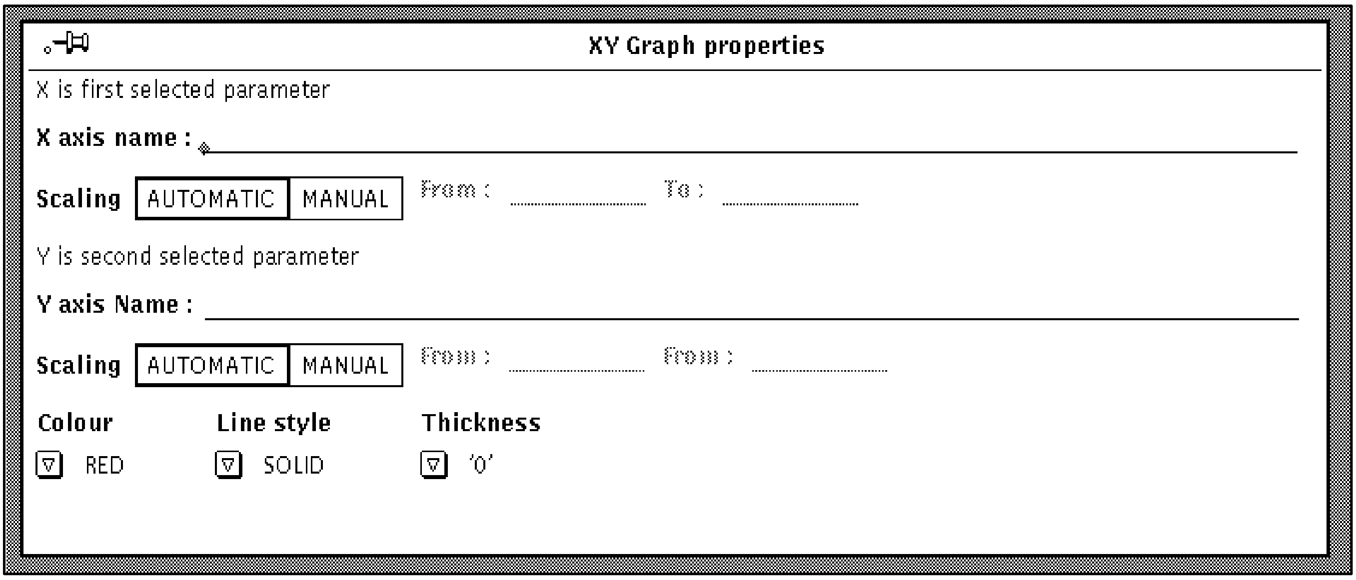


Figure 9–38 : *XY Graph properties*

The Pie Chart properties window comprises a colour menu (see Figure 9–39) to select a colour for each parameter.

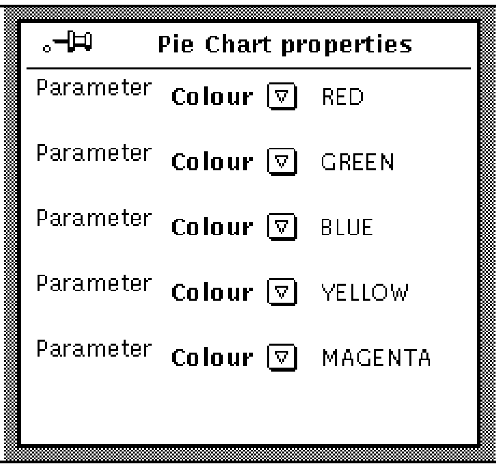


Figure 9–39 : *Pie Chart properties*

Create a Graphic

- Press the **Result...** button in the main window.
- Select the **Exec & Display** option from the pop–up menu.

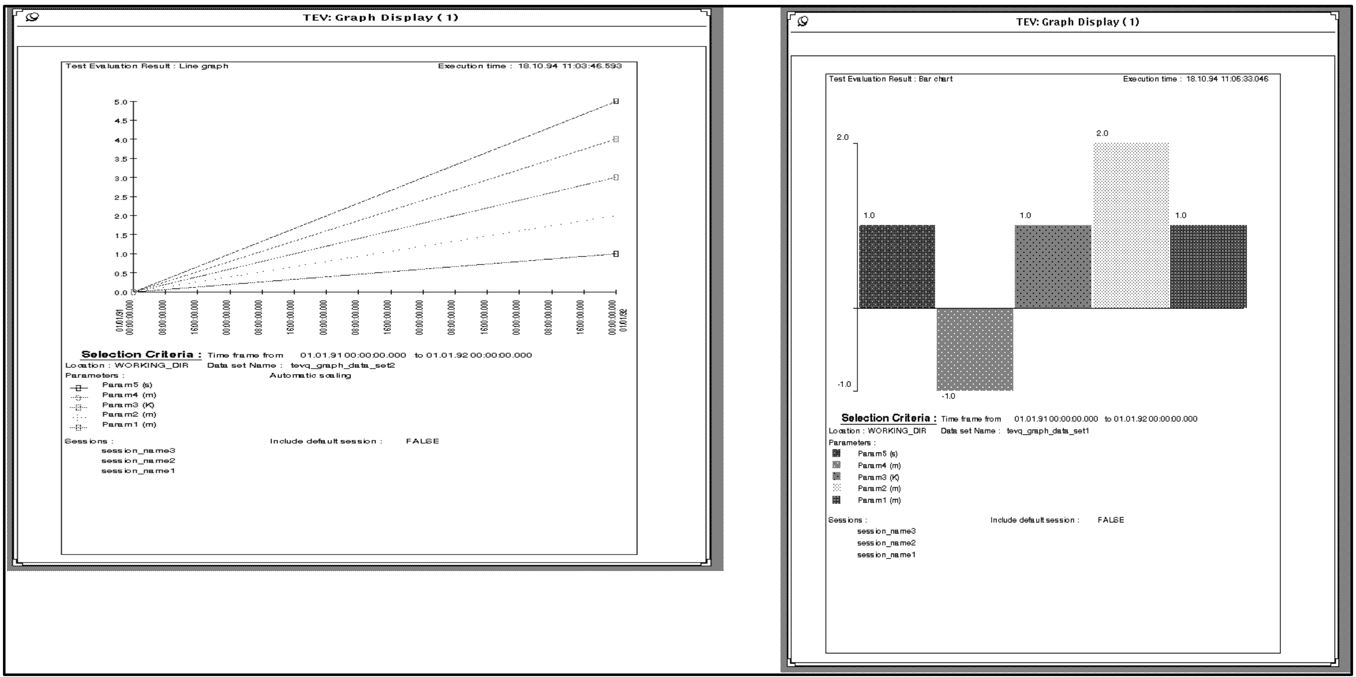


Figure 9–40 : *Examples of Line Graph and Bar Chart*

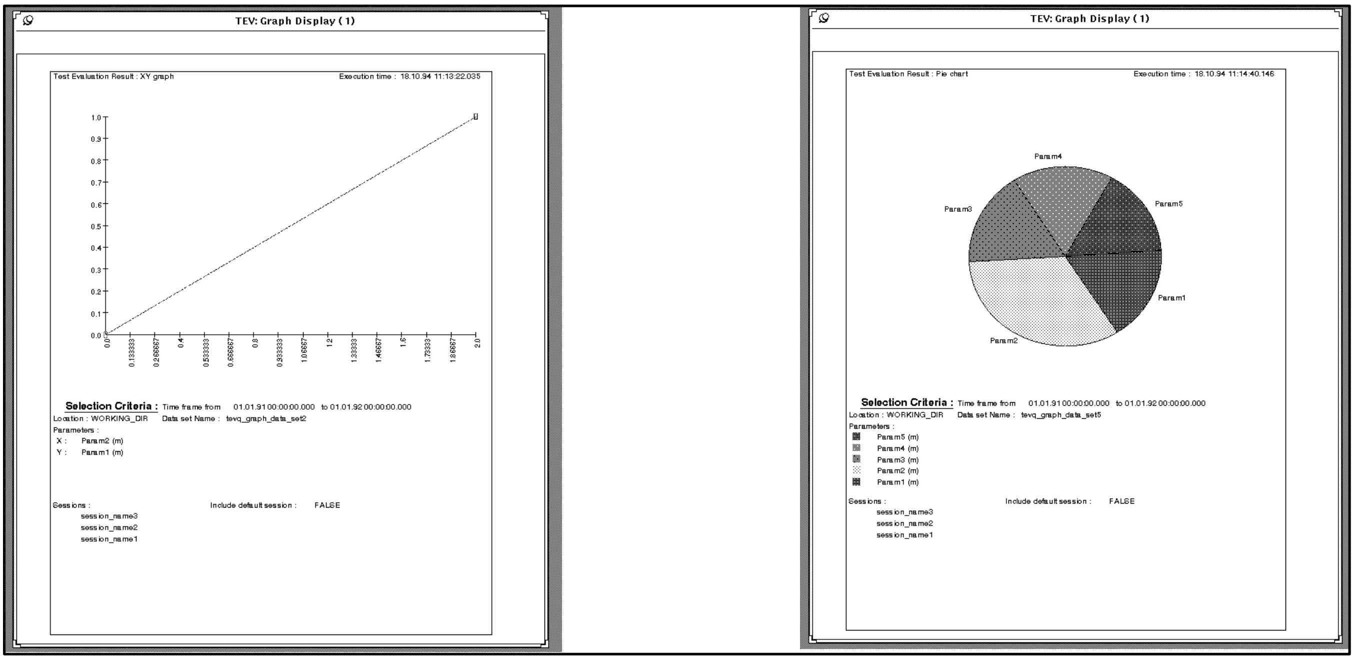


Figure 9–41 : *Example of XY Graph and Pie Chart*

9.5 Utilities

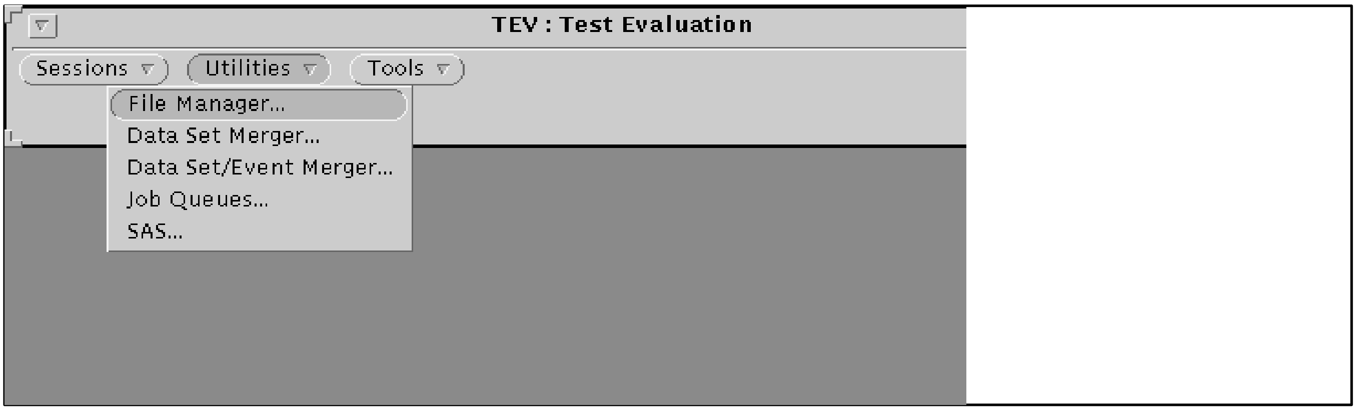


Figure 9–42 : *Utilies Menu*

The Utilities Tool offers several miscellaneous tools :

- The File Manager to manage the files which are the results of the User evaluations : Events Listing, Raw Data Dump, Data Sets, Graphs, Statistics, Data Sets Listings. These files can be deleted, renamed or copied in any directory. They can be stored in an Evaluation Session. Data Sets and Data Set Listings can be converted in Excel format.
- The Data Set Merger allows to merge two Data Sets, so that for example the data from two different sessions shall be mixed.
- The Data Set / Events Listing Merger allows to mixed events and engineering values together, ordered by time tags.
- The Job Queues Utilies allows to view and cancel print jobs and archive jobs
- The SAS Utilities allows to start SAS from TEV

9.5.1 File Handling

The Files Manager allows the user to perform operations on result files in the working directory. The working directory is located in the users UNIX file system : user_home/wd/tev/RESULTS. All the results of evaluation are per default stored in the working directory.

When the user explicitely wants to store a result file in an evaluation session, the result file is then stored in the so–called TRDB. The user has no possibility to set–up a mode, which would implicitely store all the result files in the current selected evaluation session. The user shall explicitely use the **Store** button from the **File Handling** Utilities.

To access this tool the user must select the **File Manager** entry in the ***Utilities*** menu.

The **TEV : File Manager** (Figure 9–43) window is made of :

- a **Result Type** selector. Only one tool type can be selected at a time,
- a list of file names existing in the user_home/*wd/tev/RESULTS* directory, that match the selected result type,
- a counter field *’File(s) Selected’* giving the number of selected files in the list,

- a group of five buttons: **Rename, Store, Delete, Copy** and **Convert.**

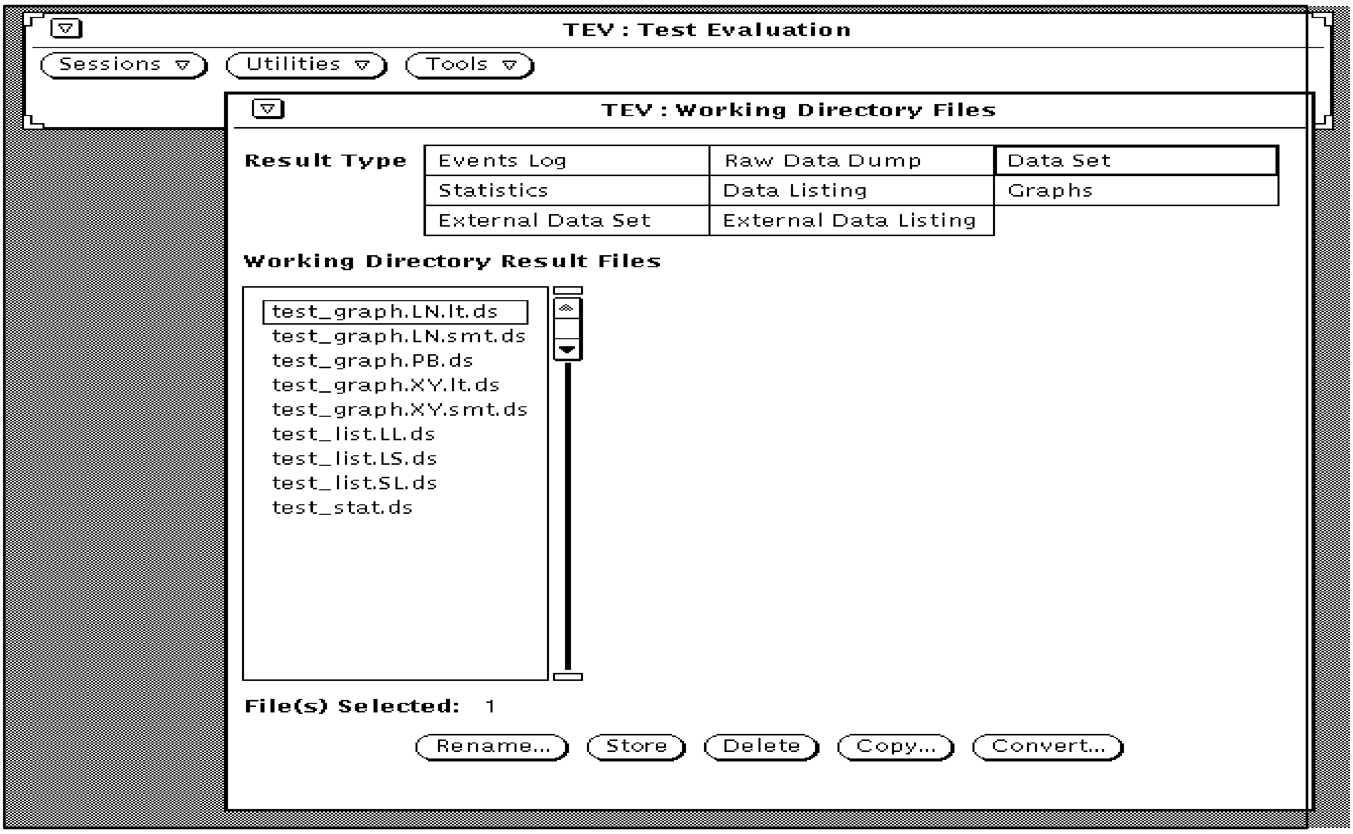


Figure 9–43 : *The working directory Files Manager*

In case the contents of the working directory or the TRDB has changed (due to work in another tool in another window) and is not up–to–date, clicking in the file type choice boxes will refresh the displayed list.

Rename and **Store** options work on single files only. **Delete** and **Copy** functions work on one or more files.

Store a file in the Test Result Database (i.e in a Test Evaluation Session)

- Perform the procedure: **Select an Evaluation session** (see Chapter 9.3.1).
- Press one of the **Result Type** buttons to determine the type of result files to be listed.
- Scroll the file list by using the scrollbars, if necessary.
- Move the mouse into the list and select the desired file.
- Press the **Store** button.

Delete one or more files from the Working Directory

- Press one of the **Result Type** buttons to determine the type of result files to be listed.
- Scroll the file list by using the scrollbars, if necessary.
- Move the mouse into the list and select the desired file(s).
- Press the **Delete** button.
- Confirm the deletion in the confirmation notice window.

The ***Rename*** operation changes the name of a file. The new name is prompted in a command window.

A default name is proposed in this case (the current name of the file). The user can edit the name in the text field; when he is satisfied with the new name, clicking on the **Rename** button will perform this operation. If the new file name corresponds to a file that already exists, a confirmation notice appears to confirm **Overwrite** or **Cancel.**

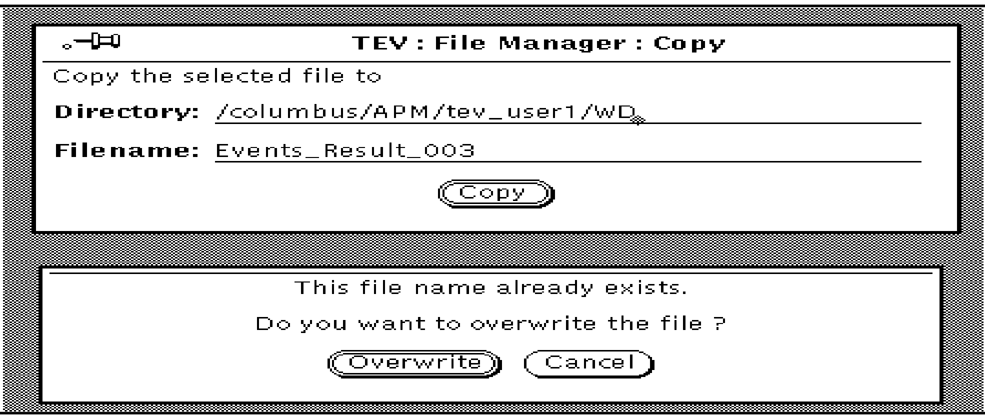


Figure 9–44 : *The command window to copy a file plus confirmation window*

Copy a single file

- Select a file name from the list.
- Click on the **Copy...** button, a pop–up window appears with two editable text fields and a button labeled **Copy** (see Figure 9–44).
- Type the destination directory in the **Directory** field.
- Enter the new filename in the **Filename** field.
- Click on the **Copy** button.
- If the given pathname corresponds to a file that already exists, a confirmation notice appears.
- Press the **Overwrite** button.

The feature to copy a single file allows to determine a new name for the file.

If you copy several files they are copied under their original names, i.e. source and destination names are the same.

Copy several files at once

- Select the result names from the list.
- Click on the **Copy...** button.
- The command window Figure 9–44 appears.
  Note that only the **Directory** field is editable.
- Enter the destination directory.Click on the **Copy** button.

The copy operates from the first (topmost) selected file to the last. If an error is encountered, the files which have been copied successfully are deselected and the ones not yet copied remain selected. The user may choose to abort the operation or retry with the files which are remaining selected.

9.5.1.1 Converting Data Set and Data Listing to Excel

The *Convert* operation is used to convert a data set or data listing file into a csv file.

When converting a Data Set, the result is sorted parameter by parameter. This means, all the values for the first parameter (measurement) are first listed, than all the values for the second parameter (measurement) and so on.

By converting a Data Listing, the result is sorted by time tags. This means the values existing for all the parameters for the first time tag are first listed , than the values for all the parameters for the second time tag and so on.

The **convert** button is only enabled when the user selects as ***Result Type*** : ***Data Set*** or ***Data Listing***.

*The user may specify which Excel separator shall be used for the conversion. The character may be setted–up into the configuration file **$TEV_HOME/config/tev_configuration.data** for the token **EXCEL_SEPARATOR**. The character **”,”** is used per default.*

After having selecting the ***Result Type : Data Set*** or ***Data Listing,*** the user may select in the list “***Working Directory Result Files***” the name of the file to convert by clicking on it, than click on the button ***Convert***. The csv file is written into a specific directory of the users working directory ie : $VICOS_TEV_WD/EXTERNAL/DATA_SET or $VICOS_TEV_WD/EXTERNAL/DATA_LIST, from which the csv file shall be copied into a directory accessible for EXCEL. To see the list of csv files the user may select external data sets or external data listing. The **store** button is disabled when the user has made this selection, because csv files cannot be stored into the TRDB.

When the user selects the ***Result Type:Data Set***, a second scrollable list is displayed, containing the Data Sets stored into the current Evaluation Session. If a selection is made in this list, then any selection in the Working Directory list is cleared, and all the buttons are disabled except for the convert. By clicking on the ***Convert*** button the user is able to convert the selected file to csv format. The converted file is written to $VICOS_TEV_WD/EXTERNAL/DATA_SET. The file in the TRDB is not affected. By selecting any other result type, the TRDB files list is removed.

To convert Data Listings results stored into an evaluation session, the user shall first re–load this file through the Data Listing tool, from the Evaluation Session (Load from TRDB) to the Working Directory.

9.5.2 Data Set Merger

The data sets merger tool allows two data sets that were built with the same time type criteria to be merged

To access the Data Set Merger, the user must select the '**Data Set Merger...**' item in the *Utilities* menu.

The **TEV : Merge Data Sets** window is shown. From this window, the user is able to merge two data sets. The user should click on the ***First*** data set choice box to select the first data set and parameters. He should click on the ***Second*** data set choice box to select the second data set. The window toggles between the two data sets.

The following cases will cause error messages :

– If the engineering units for a selected parameter which is in both data sets are not the same

– If a selected parameter in both data sets has different values for the same time tag

– If a selected parameter in both data sets has different sources ie a mix of engineering values log-books and archived files.

Once all the parameters and their time frame are selected for each data set, by pressing the **merge...** button, TEV will build a new data set file containing data extracted from both selected data sets. The user first has to give a name for this new data set file inside a pop–up window.

TEV will not allow the user to merge a data set onto itself. An appropriate error message will be given and the merge will not take place. The merge will also be refused if the resulting data set would contain more than 10 sessions.

9.5.3 Data Set / Events Merger tool

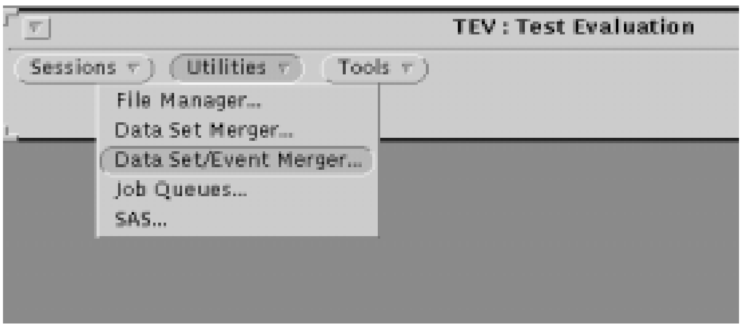


Figure 9–45 : *Start the Data Set/Events Merger*

This tool allows the user to mix measurements values which are stored in a Data Set with the events stored in an Events Listing and to order them by time tags. This tool is only thougth to handle Events Listings concerning one session only.

To access this tool the user must select the ***Data Set/Events Merger*** entry in the ***Utilities*** menu.
The window "TEV:Merge Data Set/Events" appears :

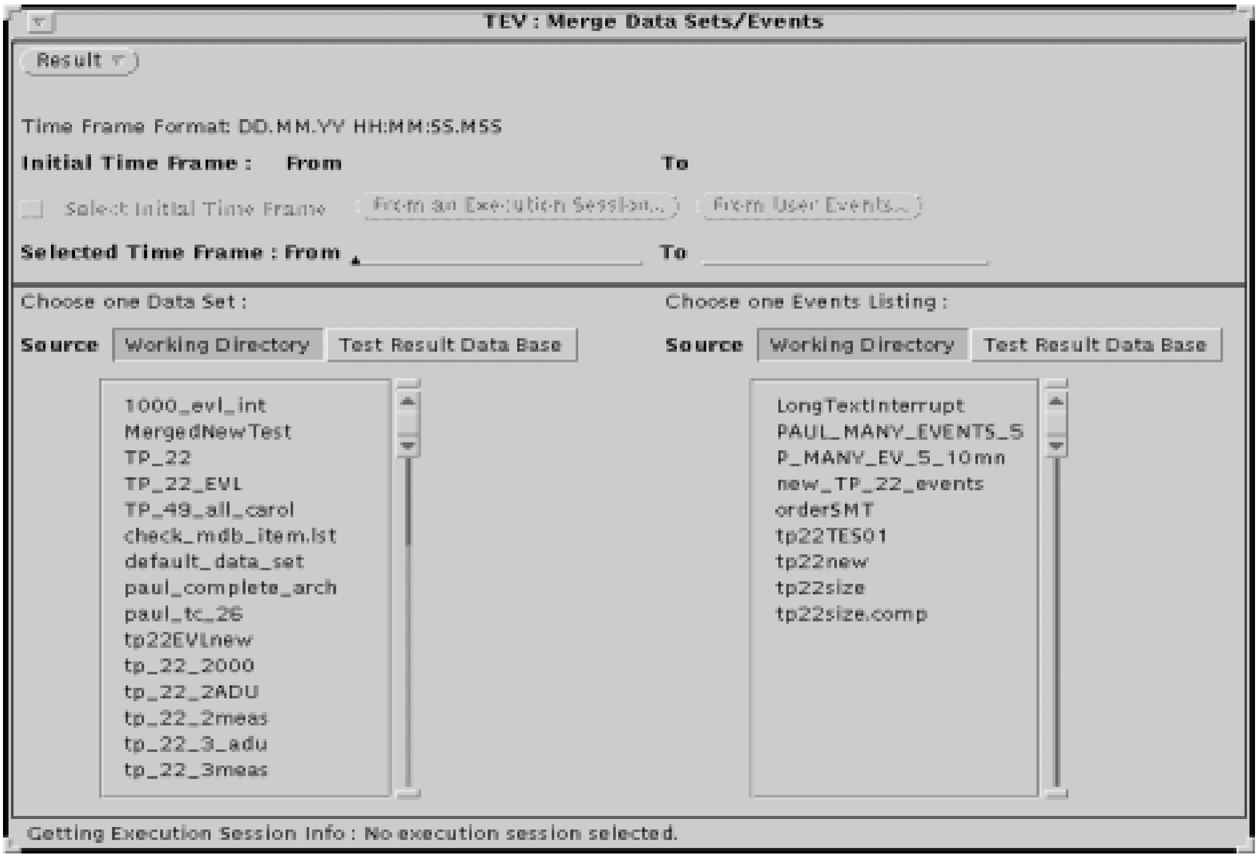


Figure 9–46 : *When no session selected .....*

If no session was previously selected, the option ***”Select Initial Time Frame”*** is not available. The ***”Selected Time Frame***” value depends on which Data Set and Events Listing have been selected
When a Data Set is selected and no Events Listing has been selected, the time frame of the Data Set appears in the ***”Selected Time Frame***” fieldst. Selecting an Events Listing when no Data Set has been selected makes appear in the ***”Selected Time Frame***” fields the time frame of the Events Listing. Selecting both shows in the ***”Selected Time Frame***” fields the combination of both time frames.

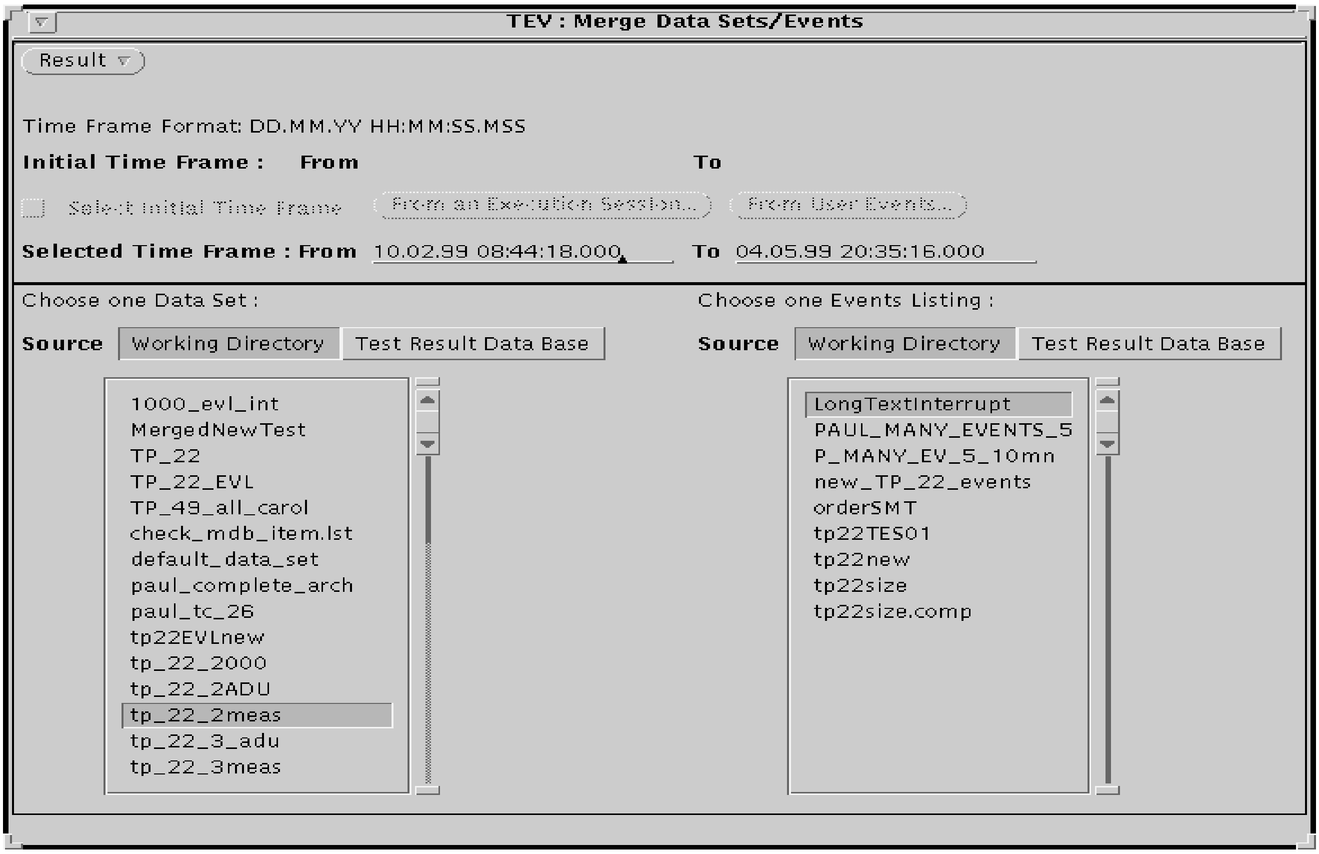


Figure 9–47 : *No session selected : Time Frame combination*

When a session has been previously selected, the user can either use the time frame of the selected session (The ***”Initial Time Frame”***) or the combination of the time frames from a Data Set and an Events Listing. In this last case, the user shall first deselect the ***Initial Time Frame*** : deselect the Choice Box ***”Select Initial Tine Frame”.*** If the Data Set and/or Events Listing were already selected before deselecting the ***Initial Time Frame,*** the Selected Time Frame is still setted–up to the ***Initial Time Frame.*** This allows the user to work on the base of the Session Time Frame. The user shall select again the Data Set and/or Events Listing in order to select explicitely the corresponding time frames.

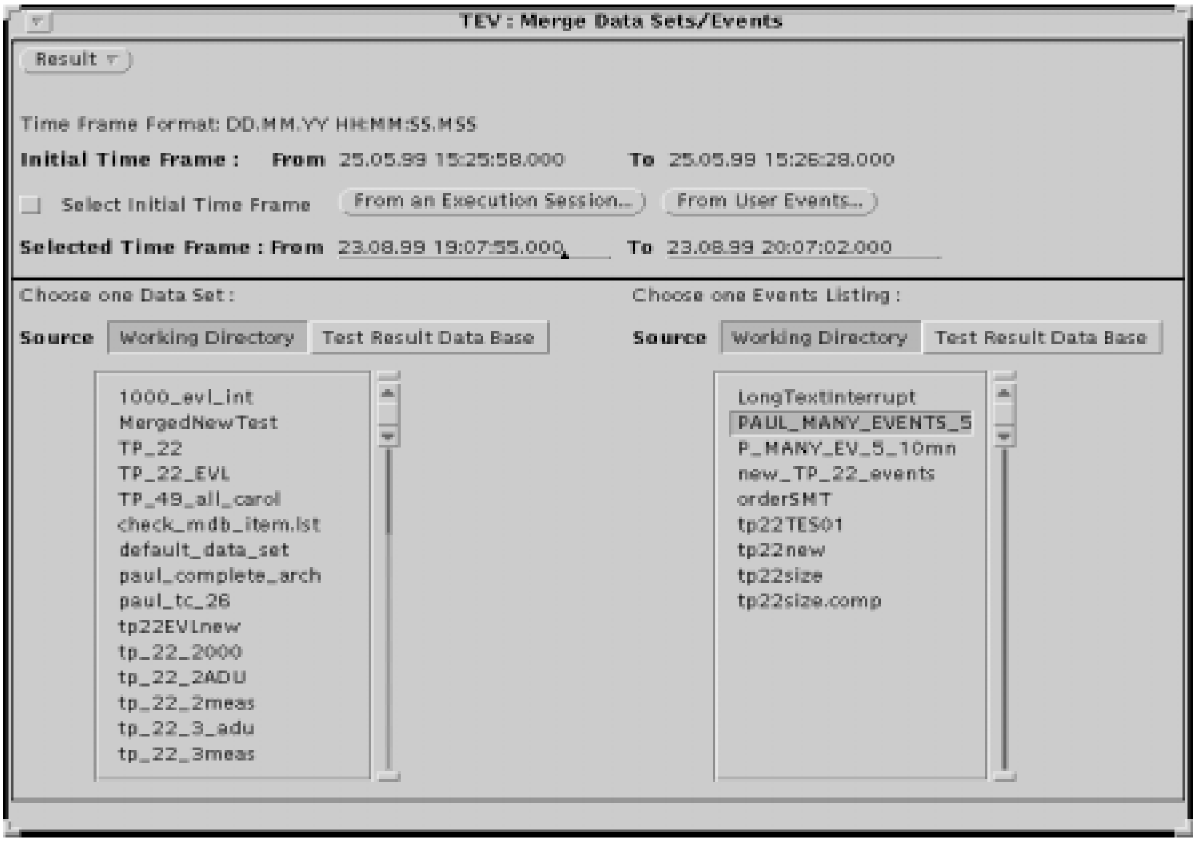


Figure 9–48 : *Session selected but Selected Time Frame from Events Listing*

Once a Data Set and an Events Listing have been selected, the user starts the merge by clicking : ***Result:Exce&Display.***
The Data Set and the Events Listing shall have been generated with the same ***Order By*** option. Otherwise the error message ***”Error: Files to merge must be ordered by same time type”*** is displayed.
The Result can be saved in the Working Directory using the menu option : ***”Result:Save”***.

To inspect a previously saved result, the user shall use the menu option : ***”Result:Load”***. The results of a Data Set/Events merging shall not be considered as a Data Sets : they can not be used for Statistics, Graph, or Data Set Listing. They can not be converted to the Excel format They can only be viewed through the Data Set / Events Merger Utilities.

Like for the Data Set Listing, the BYTE STREAM parameters are skipped (not shown) if some other types of parameters are included. If only BYTE STREAM parameters are included in the data set, then only the first one will be shown. The user has no means here to select the parameters to be shown. The Data Set shall already include the wanted parameters.

9.5.4 SAS Invocation tool

Through the SAS invocation utility the user is able to list all SAS’s in the SAS directory, supply parameters and start the SAS process. Stdout, stderr and unix messages are reported in this window.

To access the SAS Tool, the user must select the ’**SAS...**’ item in the *Utilities* menu. The **TEV : SAS Tool** window (Figure 9–49) appears. From this window, the user is able to select a SAS and to start it.

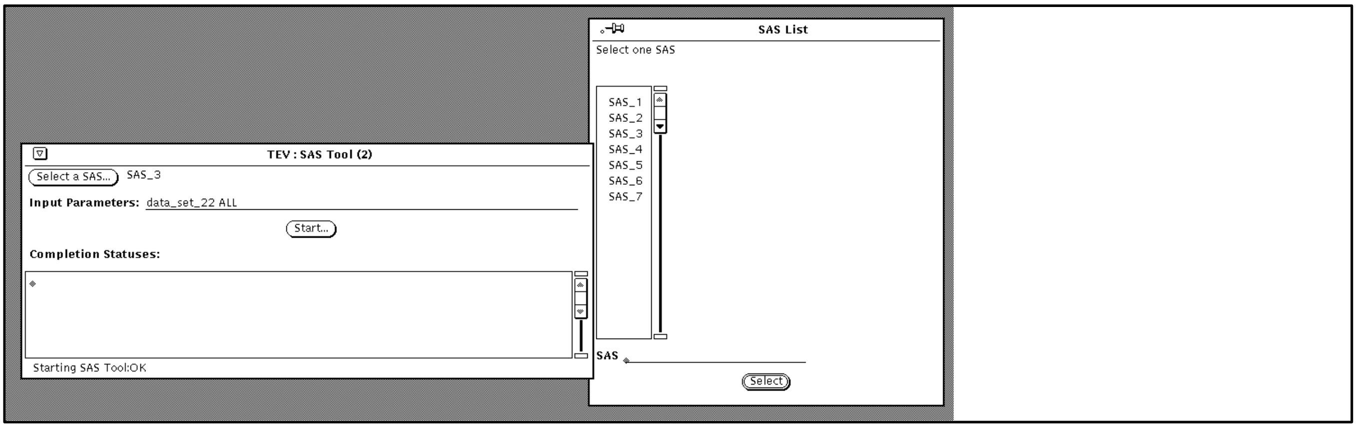


Figure 9–49 : *SAS tool window*

From the **Select SAS** button, a list of available SAS processes is displayed in a pop–up window. From this list the user can select a SAS to execute.

From the Tool window, it is possible to enter *Input parameters* which are then passed to the SAS for execution.

The selected SAS is executed by pressing the **Start** button.

Stdout, and stderr are displayed in the dedicated scrolling text zone along with Unix messages related to the SAS process.

9.5.5 Job Queue management

The Job Queue Management utility allows the user to :

- view the print jobs queues,
- select the printer to send the jobs to,
- view the final archive jobs queue,
- cancel the jobs.

To access the Job Queue Management Tool, the user must select the '**Job Queues...**' item in the *Utilities* menu. The **TEV : Job Queues** Tool window (Figure 9–50) appears.

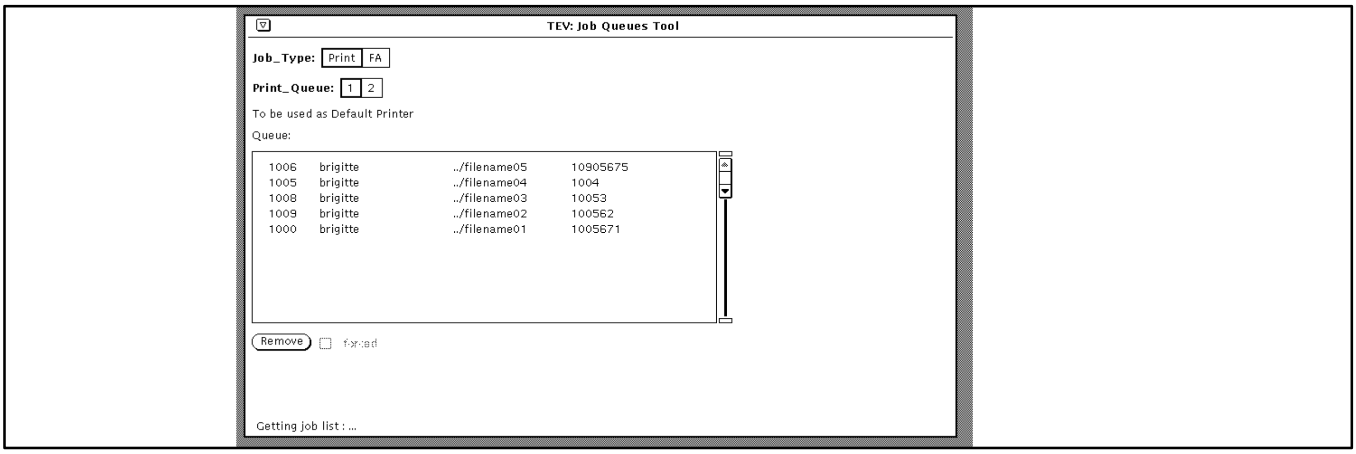


Figure 9–50 : *Job Queues Management window for print queue*

To select a queue type:

The user can view two different queues. By selecting the **Job_Type** he can choose between the Print queue or the final archive queue of the Final Archive SAS process handled by DBS. If the Print Queue is selected, the user can also select which printer he wants to access with the **Print_Queue** check box.

To select a printer queue to view:

Having selected the **Job_Type:Print**, the user can also select which printer he wants to list the queue for with the **Print_Queue** check box.

To select a printer queue to send the jobs to:

From the **Select Default Printer** choice boxes, the user can choose the printer **1** or **2** . All the further print requests will be sent to this printer.

The identification of the printers 1 and 2 is made through the $DBS_HOME/user_env/dbs_cshrc file in conjunction with parameters in the $DBS_HOME/config/dbs_configuration_file.def and the /etc/printcap system files.

To remove a job:

The user may either remove a job from the print queue or from the final archive queue. If the user is working with the print queue then the **forced** check box is disabled. If the user is working with the final archive queue then the forced check box is enabled. By selecting forced the user is able to cancel the job of another user. If the job is active then forced has no effect.

Select a job line in the list sub–window and press the **Remove** button.

*To refresh the status of the jobs queues, the user shall click again on the check boxes **Job_Type:Print** or **Job_Type:FA**.*

10 TRDB TOOLS

10.1 General

The DBS application controlling the Test Result Data Base (TRDB) proposes two additional tools :

- The Final Archive Special Application Software (FA SAS) is to be used to drive the Final Archiving medium i.e the optical disks.
- The Recovery Scripts to be used to control and manage the consistency of the TRDB in cases of errors.

10.2 Final Archive Special Application Software

10.2.1 Introduction

The following sections describe the operator interface of the Final Archive Special Application Software (FA SAS).

The FA SAS is the interface program between DBS and the Final Archive medium.

The FA SAS/DBS interface is designed in such a way that it is possible to write another FA SAS program depending on the Final Archive Medium connected (e.g. tape, magneto–optical disks) without having to modify the DBS software.

The Final Archive Medium presented in this document is made of 2 magneto–optical disk drives.

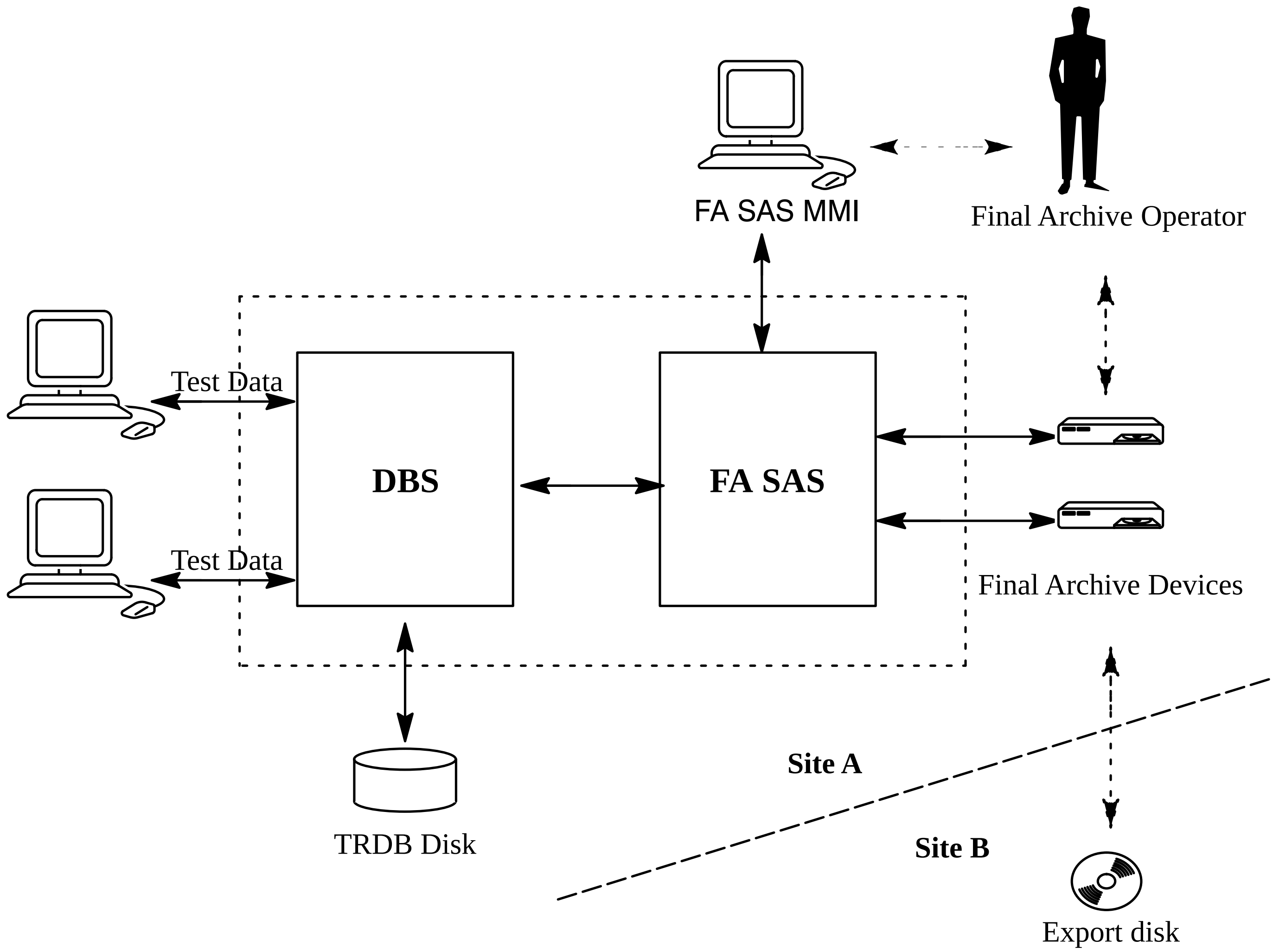


Figure 10–1 : *FA SAS Purpose*

The purpose of the FA SAS is :

To archive or retrieve sessions. In this case the FA SAS can be seen as an extension of the TRDB. When a session is archived, all the data of the session is moved (copied + deleted) from the magnetic disk of the TRDB to a magneto–optical disk. Only references are maintained in the TRDB to allow a retrieve session operation later.

To export/import a session to/from another test site via magneto–optical disks. Then it is a whole session content (data + references) which is copied to/from magneto–optical disk.

10.2.2 External View of the FA SAS

The FA SAS shows two different external views as it executes requests from **DBS** or from the **Final Archive operator**.

The FA SAS operator interface provides the following operations :

- Mount a disk in the drive,
- Enable the Ejection Button of a drive,

- Disable/Enable DBS access to the FA SAS,
- List contents of a disk,
- Print contents of a disk,
- Display the free and maximum space of a disk.

10.2.2.1 Input Data

The only data the operator may have to enter is the ***label of a disk***. This is needed during a disk mount operation when he wants to use a specific disk. All the other operator requests do not require any input data. The operator will have to select options using the proposed buttons.

10.2.2.2 Output Data

The FA SAS informs the operator about its internal states by the MMI.

A sub–window is also used to list error messages that may occur during the processing (e.g. Mount TIMEOUT).

10.2.3 Operations Environment

10.2.3.1 Hardware Configuration

The FA SAS program runs on the DBS server workstation (SUN).

One or two Magneto–Optical disk drives (e.g. Sony SMO–F521) must be connected to the DBS server workstation. The DBS server must have one or two raw disk devices available for the Final Archive SAS, i.e. none of these devices will be used by anyone else when Final Archive SAS is running

10.2.3.2 Software Configuration

The operator interface has been developed with the XVIEW library. It requires a Xwindow Server and a Window Manager that supports the OPENLOOK Graphic User Interface.

10.2.3.3 Operation Constraints

10.2.3.3.1 FA SAS must be run as 'root' super–user.

The FA SAS has to mount and dismount disks during execution. For these purposes a special script is used. As the Unix Operating System requires super–user privileges to execute these operations, this script must be run with super–user privileges. The necessary privileges are assigned to the script during CGS installation (see applicable document CGS Installation Manual [2.3.1]).

10.2.3.3.2 Disks have to be formatted.

The FA SAS expects a new disk for usage as an Export or Archive disk to be formatted and having a filesystem installed (newfs) by an off-line activity outside the FA SAS processing. It has to be made sure, that the root filesystem created is owned by the CGS administrator.

An example of formatting and installation of file system is given in the chapter 10.2.6.

10.2.3.3.3 User Interface Requests

The Final Archive program receives requests from 2 sources : DBS and the user interface.

- Some DBS requests require a manual operation (typically to insert a disk). In this case, the FA SAS program uses the user interface to send instructions to the operator. *The operator has to follow the FA SAS directives.*
- The FA SAS program assumes it is the only process to access the magneto–optical disks mounted, i.e. *the operator cannot access the disks in the drives (e.g. to mount/dismount a disk) if he is not asked or explicitly allowed to do it by the FA SAS.*

10.2.3.3.4 FA–SAS Disk Types

There is no difference anymore between Archiving/Retrieving disks and Export/Import disks. Sessions stored on MO disks are treated in the same way. The only difference between archived and exported sessions is the information that is stored and kept in the TRDB for archived sessions. No session information is stored for exported sessions. This allows the user to delete the entire archived session from the TRDB and to import it again at any time. Please note that archived sessions of older CGS versions cannot be treated as Export session and, thus, cannot be imported again in case the TRDB information is lost.
The labels of MO disks can be defined freely by the user. Nevertheless, the FA SAS will provide a default label for new disks, which can be accepted or modified by the user.

10.2.3.3.5 Virtual MO device

The FA SAS is able to handle ”virtual” MO devices, i.e. a directory located in the filesystem to be treated as MO device. A virtual device will replace a real device, i.e. only two devices can be handled by the FA SAS as a maximum. The definition of virtual devices has to be done via the FA SAS configuration file, parameter ”FA_DEVICE_FILENAME1” or/and ”FA_DEVICE_FILENAME2”. Prerequisite: the device must be block–structured or a device provided by remote machines.
Examples of a definition:

```
! block structured device
FA_DEVICE_FILENAME1 3 ”/dev/dsk/c0t0d0s7”
! device provide by remote machine
FA_DEVICE_FILENAME1 3 ”cgs–test:/cgs/users/cgsadmin/virtual_MO”
```

Please note that it is easily possible to allocate a virtual device on the same machine (database server) by defining it as remote.
A definition like

```
FA_DEVICE_FILENAME1 3 ”/cgs/users/cgsadmin/virtual_MO”
```

would cause an error on UNIX level (Message: ”not a block device”).

The advantage of virtual devices, obviously, is that single sessions can be exported into the file system and, thus, can be handled by backup services. Additionally, final archiving can be enabled while opening sessions without any real MO device connected to the database server. The entire session, then, will be stored in compressed form in the file system.
The following steps (executed as user ”root”) are necessary to connect and prepare a new harddisk to be used

as FA device:
Enter new harddisk into file /etc/vfstab, e.g. a device to be addressed via /dev/mo1:

    /dev/dsk/c0t5d0s7   /dev/rdsk/c0t5d0s7   /dev/mo1   ufs   3   yes   –

The disk will be accessible **after** reboot.
To make this new filesystem or parts of an already mounted filesystem accessible from remote, enter into file /etc/dfs/dfstab:

    share –F nfs /dev/mo1

and into file /etc/dfs/sharetab:

    /dev/mo1   –   nfs   rw

Finally, enter the command ”shareall”.

The entry in the fa_sas_configuration_file.def, then, would be:

    FA_DEVICE_FILENAME1 3 ”<hostname>:/dev/mo1”

10.2.3.3.6 Configuration Files Constraints

The configuration files for DBS and for the Final Archive SAS must be adapted to the local configuration, i.e. at least the **number of devices**, the **DBS owner** and the **device names** must be correct to ensure a correct functioning of the Final Archive SAS. An example of FA SAS configuration file is given in 10.2.7. There can be a maximum of 2 Magneto–Optical disk drives connected to the DBS server workstation.

10.2.4 Operations Basics

10.2.4.1 Introduction

10.2.4.1.1 FA–SAS Architecture

The FA SAS is made of a single Unix process. Each Magneto–Optical disk device is driven by an Ada Task and is associated to a command window (See Figure 10–2). DBS sends its requests to a third Ada task that is responsible for distributing the work to the 2 device controller tasks according to their availabilities and to a predefined strategy.

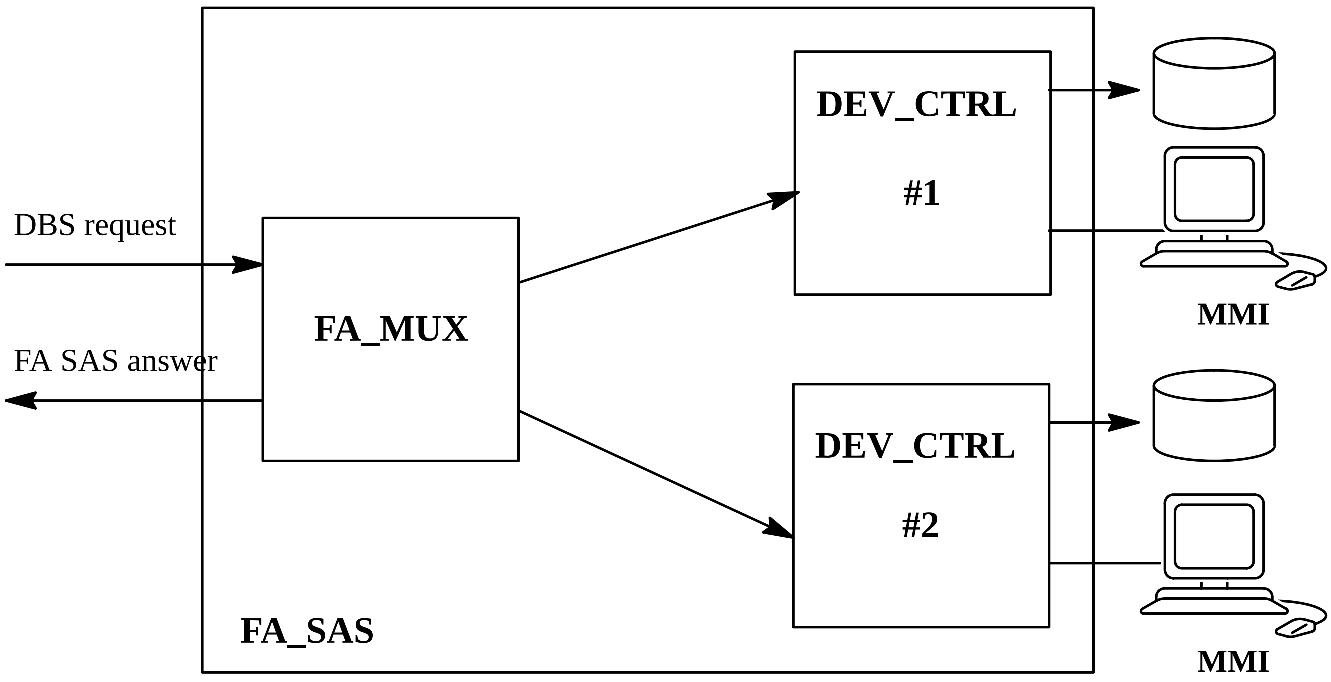


Figure 10–2 : *FA_SAS Architecture*

The FA SAS operator interface (See Figure 10–3) is made of command windows showing some basic administrative operations accessible to the operator and information about the device state (e.g. the disk currently mounted). There is one window per Final Archive device connected to the DBS server workstation. The FA SAS operator interface allows also DBS to request actions from the operator (e.g. insert a specific disk for a retrieving of data).

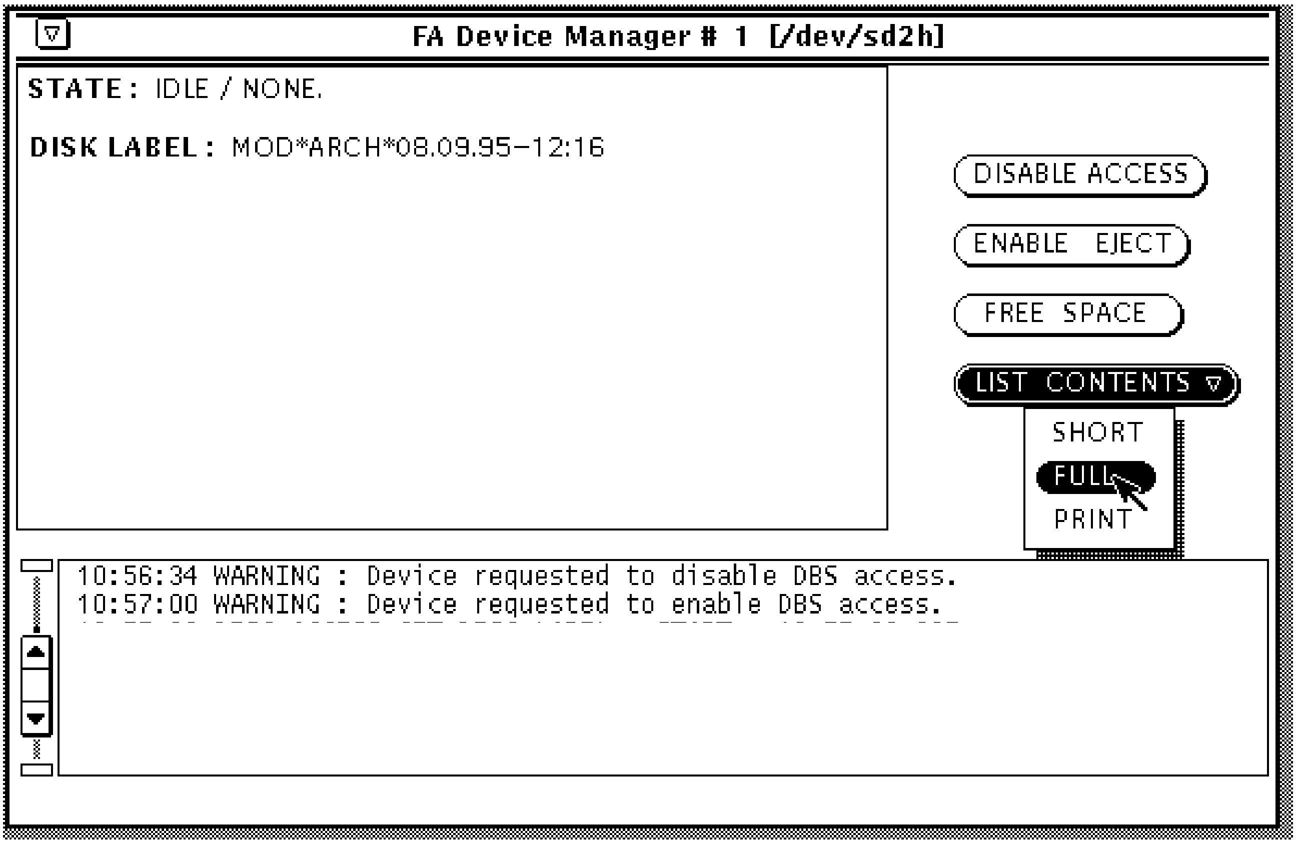


Figure 10–3 : *FA SAS Device Window – Operator Selecting the List Contents Menu*

10.2.4.1.2 FA–SAS Disk Types and FA SAS Disk Label

The FA SAS defines two disk types :

- disks for Archiving and Retrieving operations (in situ),
- disks for Export and Import operations.

A Disk Label is an information that the FA SAS writes on the magneto–optical disks to recognize them later and detect their type. As we mentioned earlier, the FA SAS requires magneto–optical disks formatted, with a file system installed and of the type corresponding to the current operation. The disk type is reflected by the FA SAS Disk Label in the way that it must contain the following prefix :

- '**MOD*ARCH***' for an Archiving/Retrieving disk and
- '**MOD*EXPORT***' for an Export/Import disk.

The FA SAS will reject the labelling of disks without these prefix. Note that the FA SAS forces the Archiving/Retrieving disk labels to values that guarantee the label unicity in the TRDB. It is not the same for Export/Import disks whose labels are not kept in the TRDB and that the operator can freely define with the restrictions on the prefix.

10.2.4.1.3 Strategy to Allocate Disk Devices

DBS has to proceed in 3 steps to obtain actions from the FA SAS. First it must request the allocation of a device for a particular purpose (Export, Manual Archive, Import, Retrieve, Automatic Archive). Secondly,

in case a device is successfully allocated for its own use, DBS must request the operation execution . Finally, DBS must request the deallocation of the allocated device. If the allocation of a device is rejected by the FA SAS, DBS can retry the same allocation later. During the time a device is allocated to DBS, the operator cannot access it by its operator interface.

The number of devices managed by the FA SAS is configurable with a maximum of 2 devices (See 10.2.3.3.6). The FA SAS allocates preferentially the device 1 to *DBS Automatic Archiving operations*, and the second device, if existing, to *DBS Manual operations* (e.g. export of session, retrieving of session data). Both devices are equally accessible for the operator administrative operations.

The FA SAS treats one request at a time on a given device. For example, during all the time it is processing a DBS request on device 1, it will not accept any other request concerning that device neither from DBS nor from the user interface.

The FA SAS assigns DBS jobs to devices according to
o their current internal state (BUSY, IDLE, ALLOCATED),
o the last operation executed by the device,
o the type of the operation to execute.

When a request for EXPORT is received by the FA SAS:

- if there is no device currently available (idle and not allocated), the request is rejected.
- if the last operation to be executed on a now idle device was an EXPORT operation, then, the incoming EXPORT operation will be automatically executed on this device (no request to the operator). This allows a set of EXPORT operations to be made on the same disk without interruption.
- if there is a device currently executing an EXPORT operation, the incoming EXPORT request is rejected.
- if there is no disk used for EXPORT which is mounted in any device, the FA SAS requests an export disk to the operator for the device 2 preferentially or for device 1 if device 2 is busy.

When a request for IMPORT is received by the FA SAS:

- if there is no device currently available (idle and not allocated), the request is rejected.
- if the last operation to be executed on a now idle device was an IMPORT operation, then, the incoming IMPORT operation will be automatically executed on this device (no request to the operator). This allows a set of IMPORT operations to be made on the same disk without interruption.
- if there is a device currently executing an IMPORT operation, the incoming IMPORT request is rejected.
- if there is no disk used for IMPORT which is mounted in any device, the FA SAS requests an import disk to the operator for the device 2 preferentially or for device 1 if device 2 is busy.

When a request for MANUAL ARCHIVE is received by the FA SAS:

- if there is no device currently available (idle and not allocated), the request is rejected.
- if the last operation to be executed on a now idle device was an MANUAL ARCHIVE operation, then, the incoming MANUAL ARCHIVE operation will be automatically executed on this device (no request to the operator). This allows a set of MANUAL ARCHIVE operations to be made on the same disk without interruption.

- if there is a device currently executing an MANUAL ARCHIVE operation, the incoming MANUAL ARCHIVE request is rejected.
- if there is no disk used for MANUAL ARCHIVE which is mounted in any device, the FA SAS requests an archive disk to the operator for the device 2 preferentially or for device 1 if device 2 is busy.

When a request for RETRIEVE is received by the FA SAS, the disk to be used is known as part of the request parameters:

- if there is no device currently available (idle and not allocated), the request is rejected.
- if there is a device currently executing a RETRIEVE operation, the incoming RETRIEVE request is rejected.
- if the last operation to be executed on a now idle device was a RETRIEVE operation using the expected disk, then, the incoming RETRIEVE operation will be automatically executed on this device (no request to the operator). This allows a set of RETRIEVE operations to be made on the same disk without interruption.
- if there is no disk used for RETRIEVE which is mounted in any device, the FA SAS requests the expected disk to the operator for the device 2 preferentially or for device 1 if device 2 is busy.

When a request for AUTOMATIC ARCHIVING is received by the FA SAS:

- if device 1 is not available (idle and not allocated), the request is rejected.
- if device 1 is available (idle and not allocated), the request is executed on device 1.

10.2.4.2 Start / Stop

The FA SAS is started by a specific startup script and stopped when DBS stops by a DBS request.

10.2.4.2.1 Operator Administrative operations

The FA SAS operator interface provides the following basic administrative operations (See Figure 10–3):

- List contents of a disk (full/short),
- Print contents of a disk,
- Display the free and maximum space of a disk,
- Enable the Ejection Button of a disk drive,
- Disable/Enable DBS access to the FA SAS to get free access to the disk.

The operator can access these administrative operations only on a device that DBS has not allocated for its own use; this means that if a device is allocated by DBS, the buttons in the corresponding device window will be automatically disabled, preventing any access to the administrative operations.

10.2.4.2.2 List contents of a disk (full/short)

The operator is able to list the contents and label of a disk inserted into a device by selecting the LIST CONTENTS menu (See Figure 10–3). This information is displayed with full details or only main points into a window according to the selection made.

10.2.4.2.3 Print contents of a disk

As the operator can list the contents of a disk, he can print it with full details by selecting the LIST CONTENTS menu and its PRINT option (See Figure 10–3).

10.2.4.2.4 Display the free and maximum space of a disk

The operator can ask the FA SAS to display the free and maximum space of the disk inserted in a device by pushing the FREE SPACE button. A graphical gauge showing the space used on the current disk is also displayed.

10.2.4.2.5 Enable the Ejection Button of a disk drive

The operator may want to dismount a disk for several reasons. For example, the operator could insert a disk into an available disk drive to see how much free space is left on it, or to list its contents. After he got the information, the operator doesn't need that disk to be mounted anymore. The operator may also want to export a set of sessions on different Export/Import disks; if he does not eject the disk between session exports, the FA SAS exports the whole set of sessions on the same disk as explained in section 10.2.4.1.3.

The effect of this command is to enable the ejection button on the disk drive and to allow manual ejection of the mounted disk.

10.2.4.2.6 Disable/Enable DBS access to the FA SAS

The operator has the possibility to inhibit all the DBS access to a Final Archive device. This functionality is provided to 'disconnect' a disk drive from DBS. The operator is then able to use the disk drive for another application (e.g. to perform a backup, to get free space) without any DBS interaction.

The operator has the possibility to 'reconnect' the disk drive to DBS by re–pushing the Disable/Enable button.

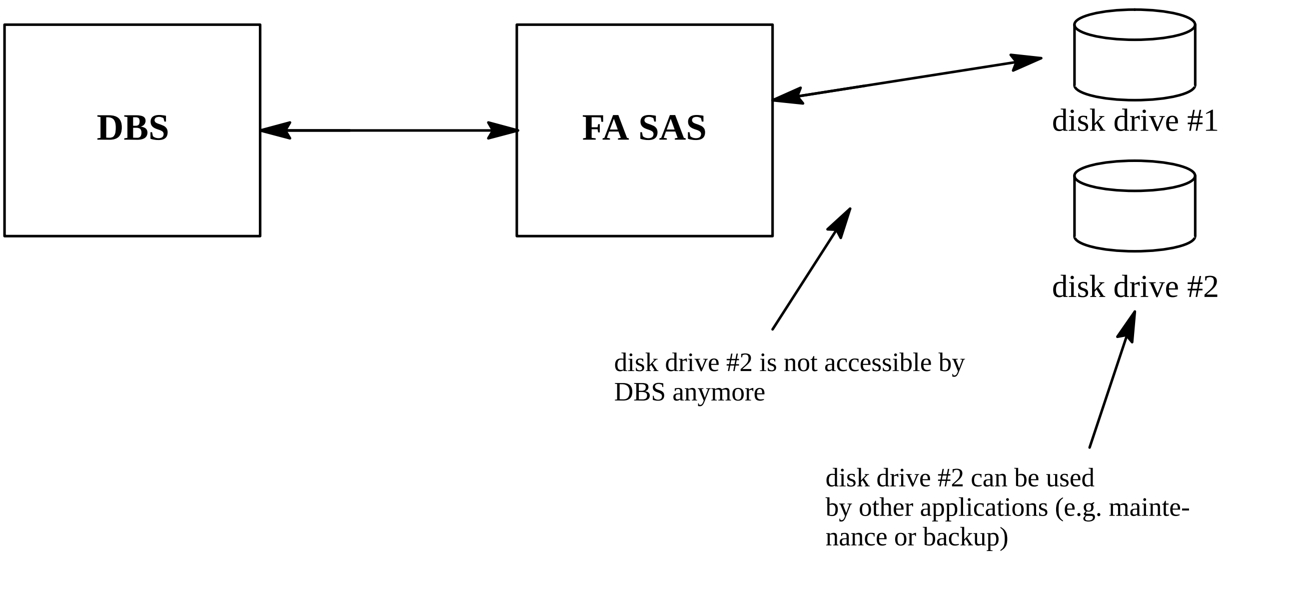


Figure 10–4 : *DBS Access Disabled*

10.2.4.3 DBS Requests to the Operator

Following a DBS operation, the operator may be required via the FA SAS MMI to insert a particular disk into a drive.

For a retrieve operation, the FA SAS knows which disk contains the data to retrieve and it gives the label of the disk that must be mounted.

For an archiving or an export operation, the operator is free to mount the disk he wants to use. If it is a new one, not yet labeled by the FA SAS, the operator will have to confirm its labelling. If it is not a new disk, the FA SAS will check that the mounted disk can be used for the required operation and request another one if not.

For an import operation, the FA SAS does not know the disk that will be imported as it is coming from another site. The operator, in this case, is free and responsible to mount the disk he wants to use.

10.2.5 Operator’s Manual

10.2.5.1 Set–Up and Initialisation

The FA SAS is started by a specific startup scripts (start_fa_sas).

Before starting the FA SAS, 4 parameters of the configuration file (fa_sas_configuration_file.def) have to be adapted to the local needs:

- The name of the device file to be used (e.g. /dev/mo1 or /dev/mo2 – this devicename has been chosen in the scope of CGS for simplification purposes. The devicename defined in the configuration file is simply a link to the ”real” hardware address of the final archive device. In case of any modification to the system only the link has to be reset but the configuration file stays unchanged).
- The number of Magneto–Optical disk devices available (1 or 2); that number must be consistent with the one defined in the DBS configuration file,
- The name of the DBS owner,
- A time-out value corresponding to the maximum number of seconds an operator can wait to answer an FA SAS request.

The startup scripts will automatically set the environment variables needed to start the FA SAS, and execute a minimal checking of the environment.

10.2.5.2 Getting Started

After startup, the Main menu window is displayed as shown in Figure 10–5.

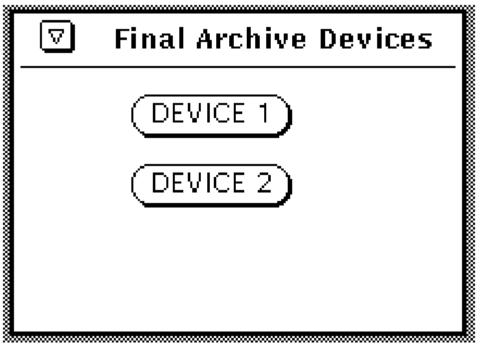


Figure 10–5 : *FA SAS Main Window – 2 Devices Available*

The main menu window contains as many buttons as devices defined in the configuration file. If the Operator click on one of these buttons, the FA SAS will display the corresponding device control window as shown in Figure 10–6.

Each device control window is made of 5 areas (Cf. Figure 10–6):

- ***TITLE*** : The number following the ’#’ character may be 1 or 2. It gives the ID number of the Magneto–Optical disk drive associated with the window. A ’1’ means that the drive asso-

ciated will be used preferentially for the 'automatic archiving operations'. A '2' means that this window is associated to the second Magneto–Optical disk drive which will be used preferentially for all the 'manual operations'. The filename between brackets is the name of the device used to access the Magneto–Optical disk drive.

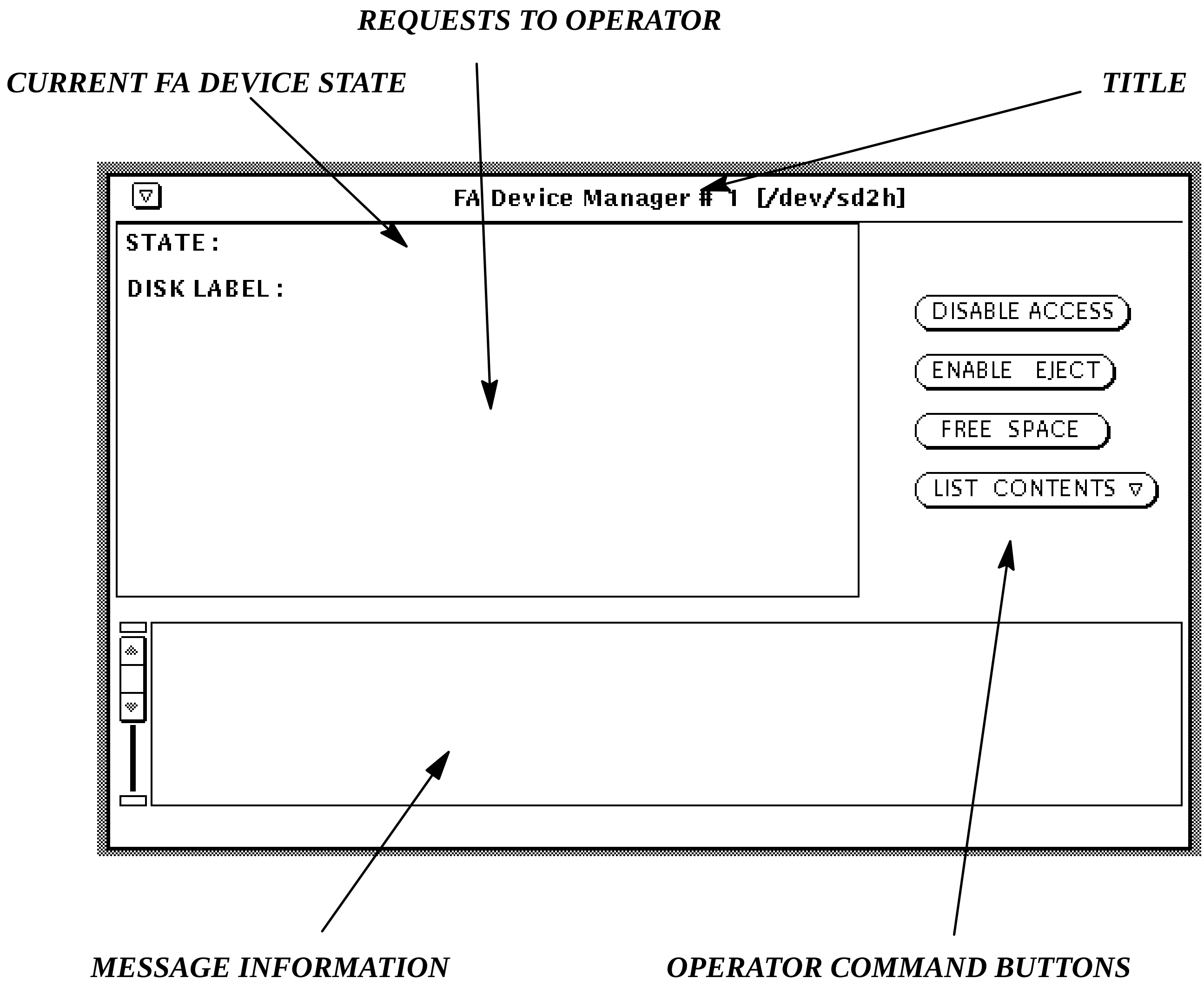


Figure 10–6 : *FA SAS Device Control Window*

- ***CURRENT FA DEVICE STATE*** : This part of the window provides general information about the current state of the FA SAS program and of the disk drive.
  - **STATE** shows the current internal state of the FA Device : the current activity (BUSY, IDLE or DISABLED), the allocation state (ALLOCATED) and the kind of operation performed on the disk (AUTOMATIC_ARCHIVE, MANUAL_ARCHIVE, RETRIEVE, EXPORT, IMPORT or NONE)
    At startup the **STATE** field contains "IDLE / NONE" and means that there is no disk mounted in the drive associated and that it is not allocated by DBS for any operation.
    A display of "BUSY / ALLOCATED for EXPORT" means that the FA device associated

is currently exporting data on the mounted disk.
A display of ”IDLE / MANUAL_ARCHIVE” means there is an archive disk mounted in the drive but that it is not used for the moment by DBS.
A display of ”IDLE / ALLOCATED for IMPORT” means that FA SAS reserved the device for DBS before starting an import operation.

- **DISK LABEL** contains the label of the disk that is loaded in the corresponding disk drive.

- ***REQUESTS TO OPERATOR*** : This area is used by the FA SAS to send requests to the operator (e.g. to ask the operator to mount a specific disk for a retrieve operation). More details are given in Chapter 10.2.5.4.
- ***OPERATOR COMMAND BUTTONS*** : The four buttons of this area can be used by an operator to disable the DBS access to the FA SAS, to enable ejection of a disk, to display free space of the current disk or to list its contents. More details are given in Chapter 10.2.5.3.
- ***MESSAGE INFORMATION*** : This subwindow is used by the FA SAS program to display messages during the processing. These messages may be errors (e.g. Disk device is full) or just information for the operator (e.g. FA device requested to stop activity). Each message contains the current date and time. The messages are not deleted automatically from the subwindow, but the operator can remove them using the OpenWindows functionalities. More details are given in Chapter 10.2.5.5.

10.2.5.3 Operator Command Buttons

The operator can access 4 command buttons on each device control window when DBS has not allocated the associated device for its own use. These buttons give access to basic administrative operations.

10.2.5.3.1 DISABLE ACCESS / ENABLE ACCESS buttons

When the DISABLE ACCESS button is pushed, DBS is not allowed anymore to access the corresponding device. The 'disabled FA device' does not accept any request coming from DBS and the FA SAS multiplexer will try to send the DBS request to the other device; another application may access the disabled device (e.g. to perform a backup) or the operator can execute other administrative commands provided by the FA SAS (e.g. list the contents of disks).

After it is pushed, the ***DISABLE ACCESS*** button is replaced by an ***ENABLE ACCESS*** button, to reset DBS access to the disk drive (Figure 10–7).

The STATE field indicates that the FA device is DISABLED and that there is no disk mounted in the drive (no disk recognized by the FA SAS program); the user can eject it or insert a new one. The usage of the other operator buttons is still allowed.

The information subwindow displays a warning message that will also be recorded in the FA SAS log file (See section 10.2.5.5).

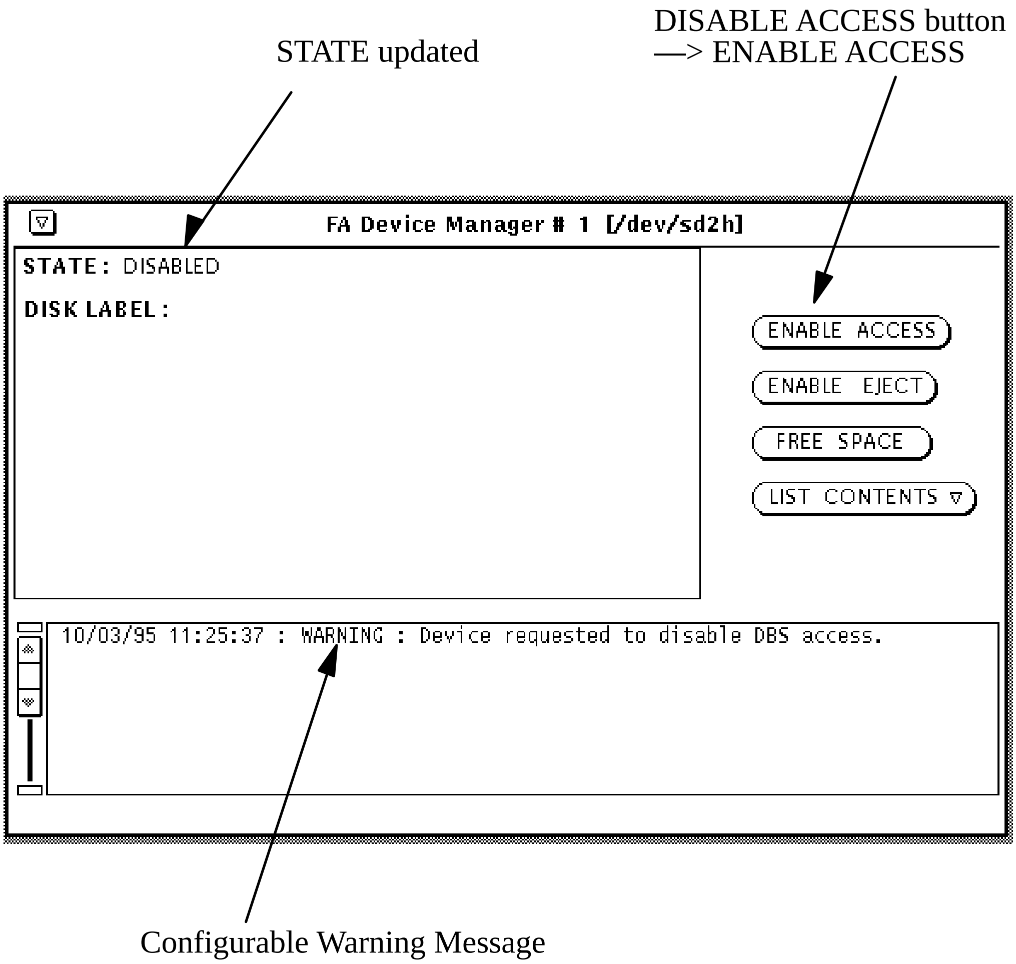
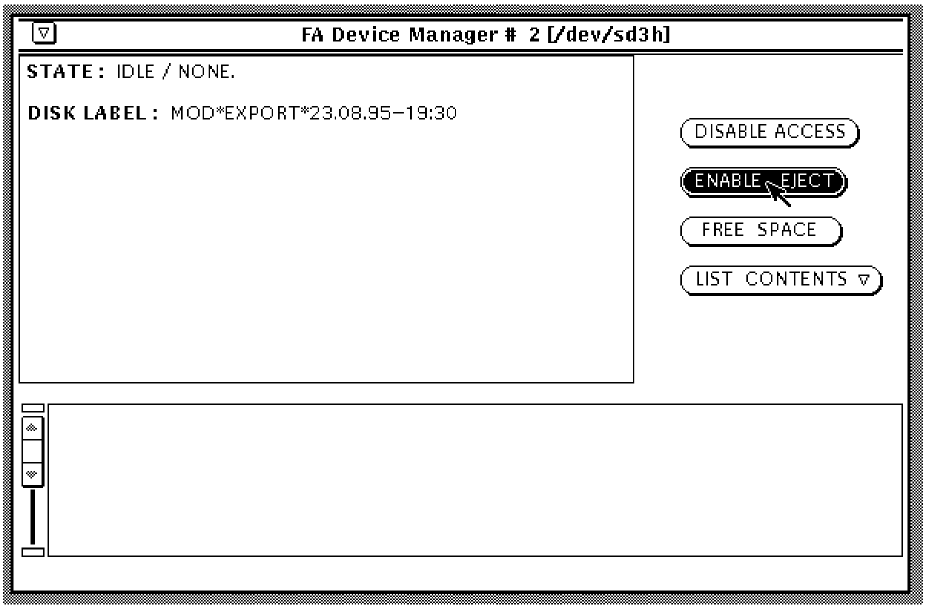


Figure 10–7 : *FA Device 1: DBS Access Disabled*

10.2.5.3.2 ENABLE EJECT button

The ENABLE EJECT button can be used by the operator when he wants to remove a disk from the drive. The FA SAS program will dismount the current disk loaded and re–enable the hardware eject button of the drive, allowing the operator to push it to eject the disk (See Figure 10–8).

1. Click on the '**Enable Eject**' button



2. Push the hardware eject button

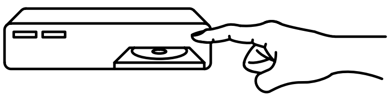


Figure 10–8 : *Manual Ejection of a Disk*

10.2.5.3.3 FREE SPACE button

The operator can get the *disk label*, the *free space* and *maximum space* available on the current disk by clicking on the FREE SPACE button. The space already used on the current disk is also displayed into a Gauge as shown in Figure 10–9.

To remove the Free Space display of the device window, the operator has to click on the ’DONE’ button. If he does not click on the ’DONE’ button, the display will disappear after a configurable period is elapsed. The operator buttons are disabled until free space display is removed.

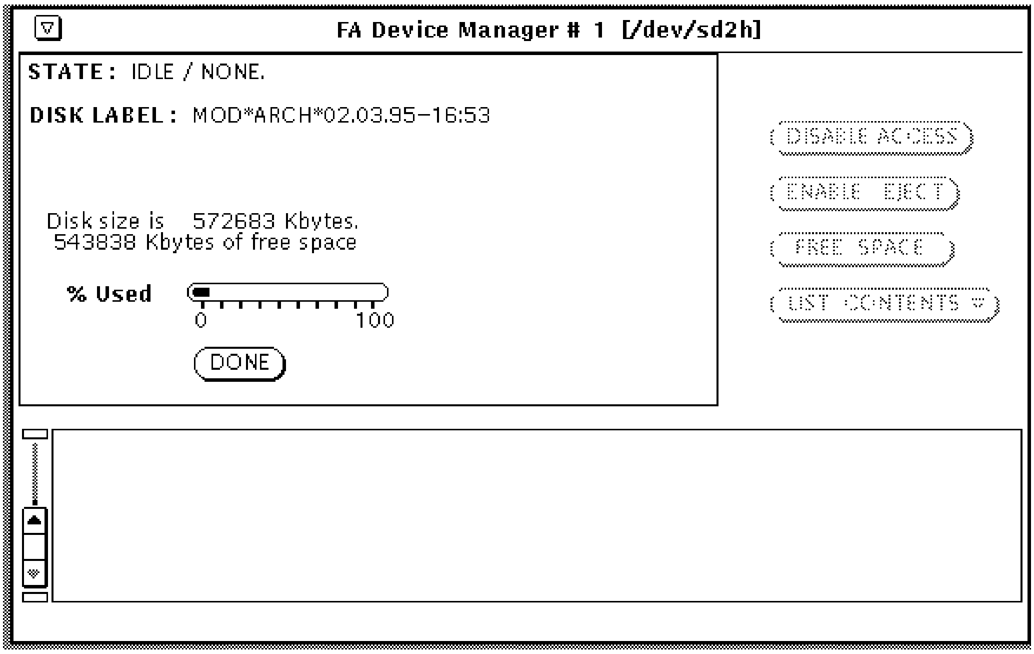


Figure 10–9 : *FA Device 1 : Free Space display.*

10.2.5.3.4 LIST CONTENTS button

A menu is associated to the LIST CONTENTS button as shown by Figure 10–10.

Select the SHORT option to generate a window containing a list of the components that are on the mounted disk (e.g. the names of the exported sessions).

Select the FULL option to generate the same list with the component structure detailed (e.g. the exported sessions with data types and all the files exported).

Select the PRINT option to send the full component structure detailed to the FA SAS printer (defined by an environment variable whose name is contained in the configuration file, Cf. 10.2.7).

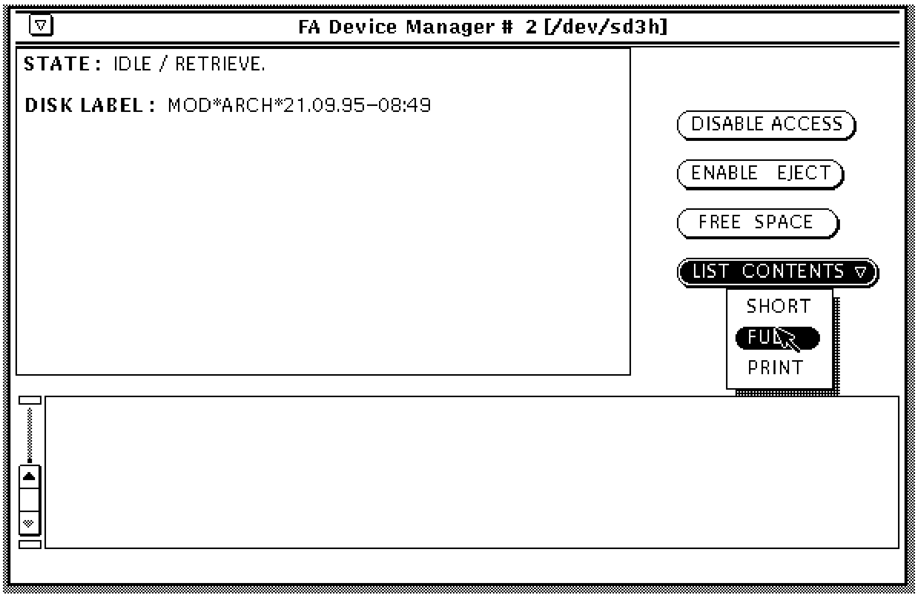


Figure 10–10 :*an FA SAS Device window, operator selecting the list contents menu*

On the figure, the LIST CONTENTS button has just been pressed, and the FULL option is selected. There is an archive disk in the drive and the FA SAS is not performing any archive operation for the moment.

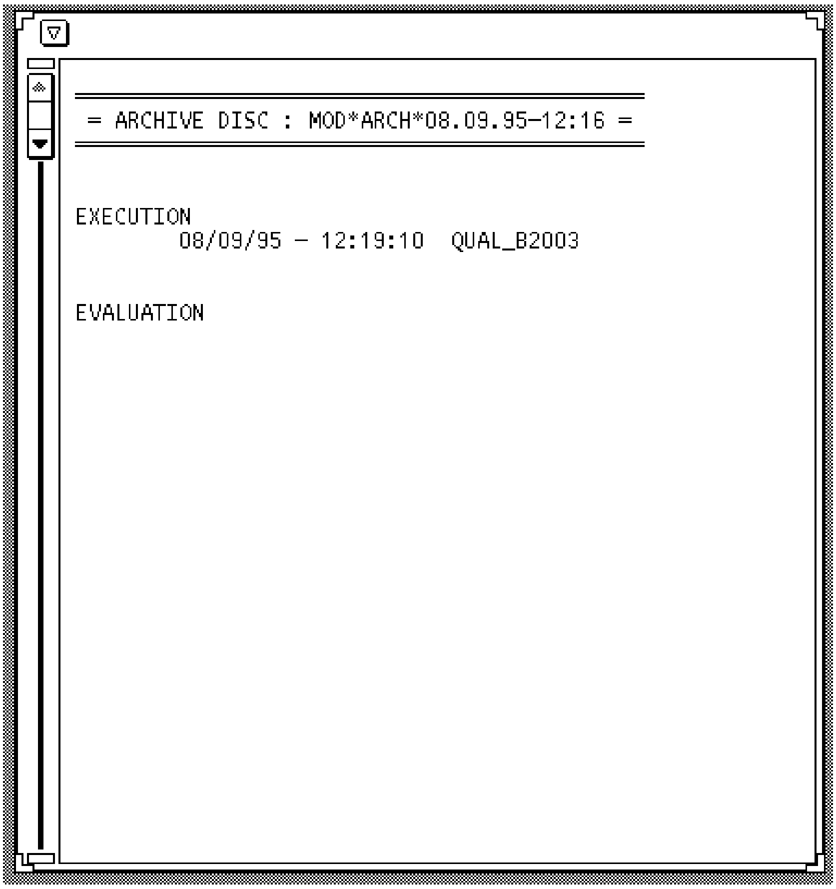
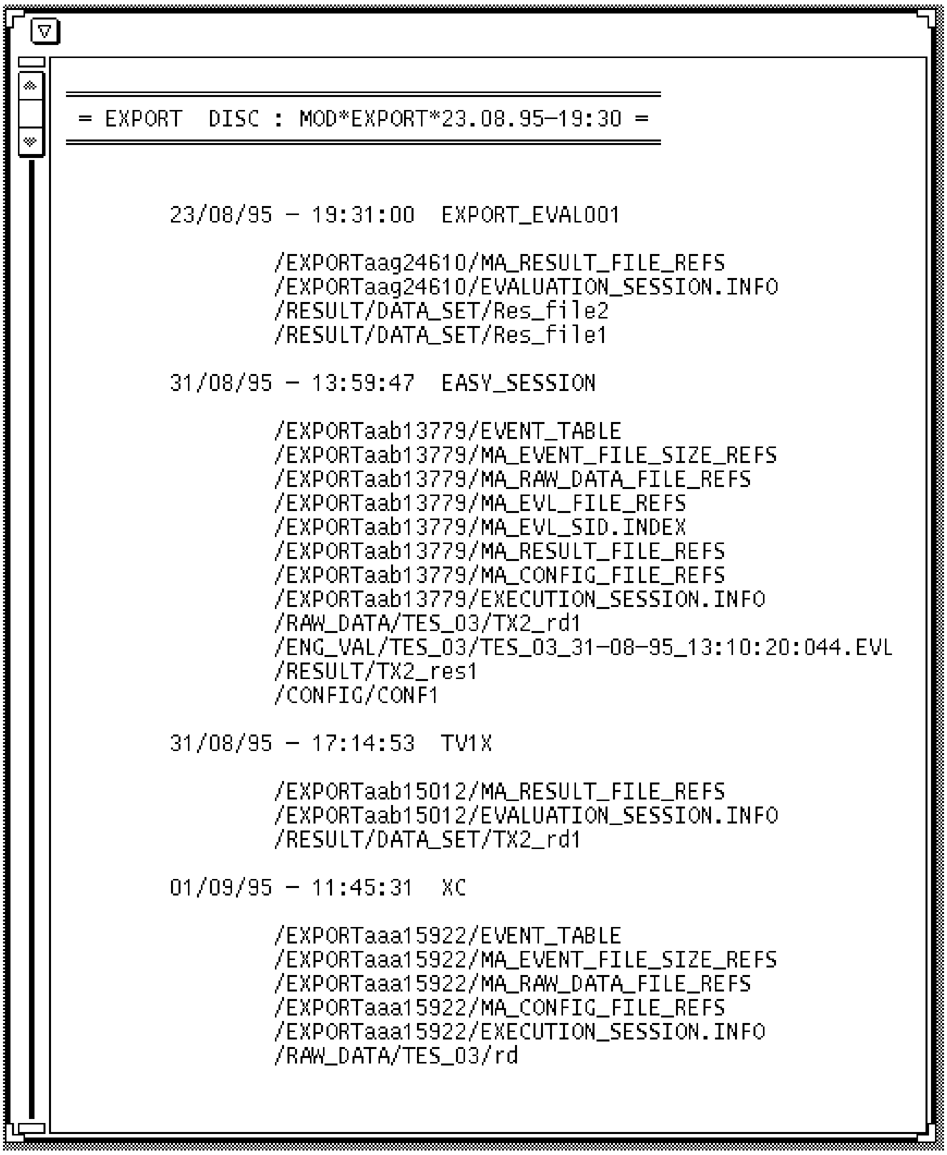


Figure 10–11 :*SHORT LIST display*

The Figure 10–11 shows the window generated by the LIST SHORT request. The listed disk is an archiving disk with the label 'MOD*ARCH*08.09.95–12:16' that contains files belonging to the test execution session 'QUAL_B2003'. No evaluation session has already been archived on this disk.



```
= EXPORT  DISC : MOD*EXPORT*23.08.95–19:30 =

23/08/95 – 19:31:00  EXPORT_EVAL001

        /EXPORTaag24610/MA_RESULT_FILE_REFS
        /EXPORTaag24610/EVALUATION_SESSION.INFO
        /RESULT/DATA_SET/Res_file2
        /RESULT/DATA_SET/Res_file1

31/08/95 – 13:59:47  EASY_SESSION

        /EXPORTaab13779/EVENT_TABLE
        /EXPORTaab13779/MA_EVENT_FILE_SIZE_REFS
        /EXPORTaab13779/MA_RAW_DATA_FILE_REFS
        /EXPORTaab13779/MA_EVL_FILE_REFS
        /EXPORTaab13779/MA_EVL_SID.INDEX
        /EXPORTaab13779/MA_RESULT_FILE_REFS
        /EXPORTaab13779/MA_CONFIG_FILE_REFS
        /EXPORTaab13779/EXECUTION_SESSION.INFO
        /RAW_DATA/TES_03/TX2_rd1
        /ENG_VAL/TES_03/TES_03_31–08–95_13:10:20:044.EVL
        /RESULT/TX2_res1
        /CONFIG/CONF1

31/08/95 – 17:14:53  TV1X

        /EXPORTaab15012/MA_RESULT_FILE_REFS
        /EXPORTaab15012/EVALUATION_SESSION.INFO
        /RESULT/DATA_SET/TX2_rd1

01/09/95 – 11:45:31  XC

        /EXPORTaaa15922/EVENT_TABLE
        /EXPORTaaa15922/MA_EVENT_FILE_SIZE_REFS
        /EXPORTaaa15922/MA_RAW_DATA_FILE_REFS
        /EXPORTaaa15922/MA_CONFIG_FILE_REFS
        /EXPORTaaa15922/EXECUTION_SESSION.INFO
        /RAW_DATA/TES_03/rd
```

Figure 10–12 :*SHORT LIST display*

The Figure 10–12 shows the window generated by the LIST FULL request. The listed disk is an EXPORT disk with the label ’MOD*ARCH*23.08.95–19:30’ that contains files belonging to 4 exported sessions ’EXPORT_EVAL001’, ’EASY_SESSION’, ’TV1_X’ and ’XC’.

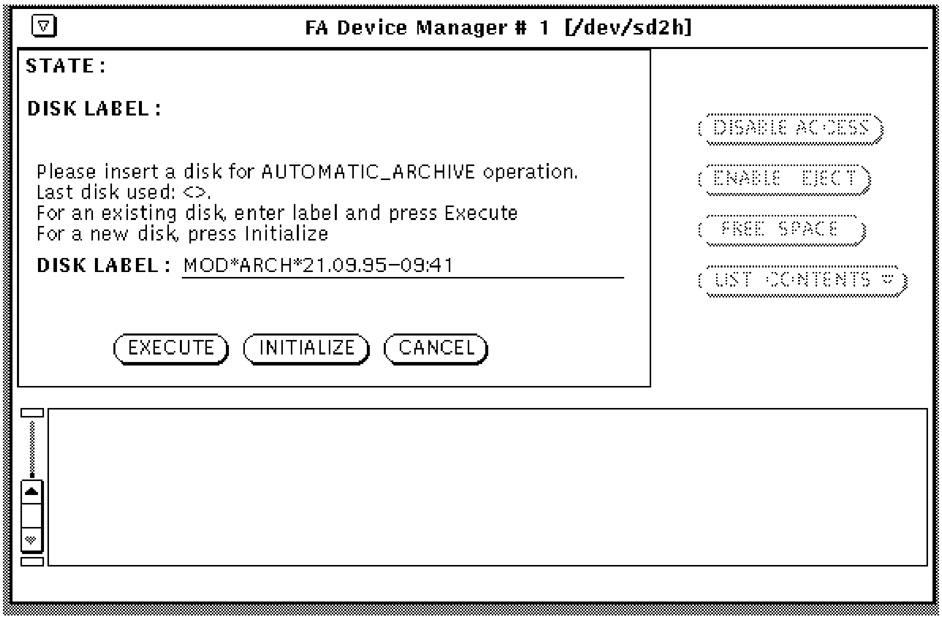
10.2.5.4 Management of DBS Requests

DBS can use the FA SAS for the following purposes:

- the ”Manual Archiving” of a session,
- the ”Retrieving” of a session or of session data,
- the ”Export” of a session or of session data,
- the ”Import” of a session,
- the ”Automatic Archiving” of an execution session.

According to the DBS request, the FA SAS may want to access a specific disk or disk type. It will display a message in the operator interface requesting the insertion of an appropriate disk (see Figure 10–6, area ”Request to Operator”). During a disk request, the hardware eject button is always enabled to allow disk changes by the operator.

The Figure 10–13 shows the message displayed for an Automatic Archiving disk request.



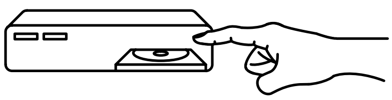


Figure 10–13 :*Disc Request for an Auto–Arch processing, the hardware eject button is enabled.*

The next chapters present the requests to the operator classified by DBS operation.

10.2.5.4.1 Manual Archiving to the FA Medium

If a node issues an Archive Session command, the FA SAS will try to allocate an FA device according to the strategy defined in section 10.2.4.1.3.

If no Archiving/Retrieving disk is available, the FA SAS will display a message requesting insertion of an appropriate disk, as shown in Figure 10–14. A time-out is associated to this disk request. Its value is a configurable parameter of the FA SAS program (See 10.2.7).

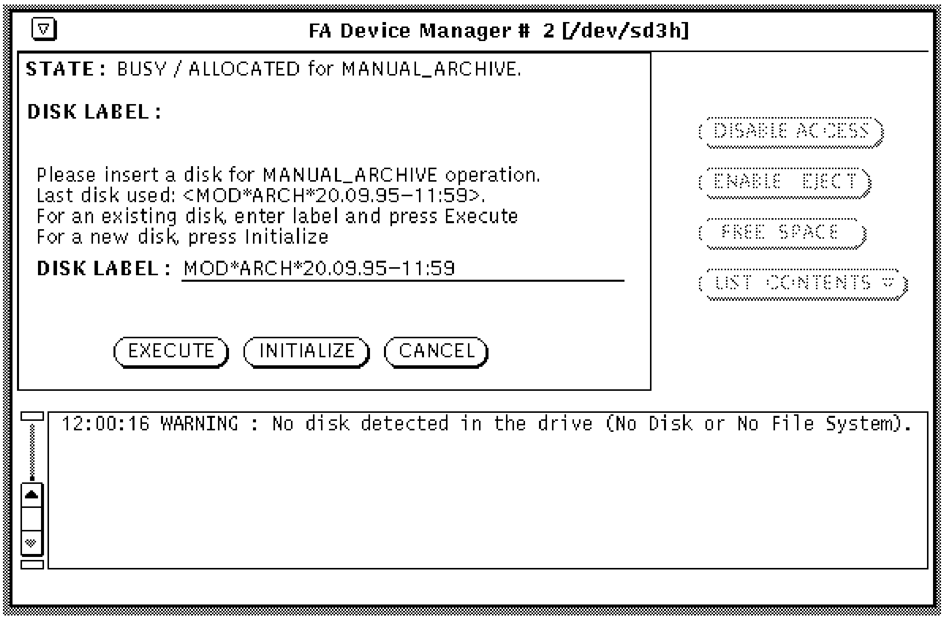


Figure 10–14 :*Disc Request for a Manual Archiving processing.*

The device state is ”BUSY/ALLOCATED for Manual Archiving”, and the buttons for administrative operations are disabled. A default disk label is displayed and the hardware ejection button is enabled to allow disk changes by the operator.

The operator may want to use a new disk, an old Archiving disk, or to re–initialize an old one. He can also cancel the operation, leading to a failure of the Archive request.

- For a new disk, or to re–initialize an old one (remove data and create new label), the operator must click on the INITIALIZE button. The inserted disk will be initialized to the default label proposed by the FA SAS after confirmation by the operator (Figure 10–15).
  If the operator cancels the initialization at this stage, the FA SAS returns back to the disk request display. Note in the confirmation display (Figure 10–15) that the initialization label is always the default one, even if the operator has previously entered another label. The FA SAS forces the Archiving/Retrieving disk label to values that guarantee the label unicity in the TRDB. As we will see, it is not the same for Export/Import disks whose labels are not kept in the TRDB.

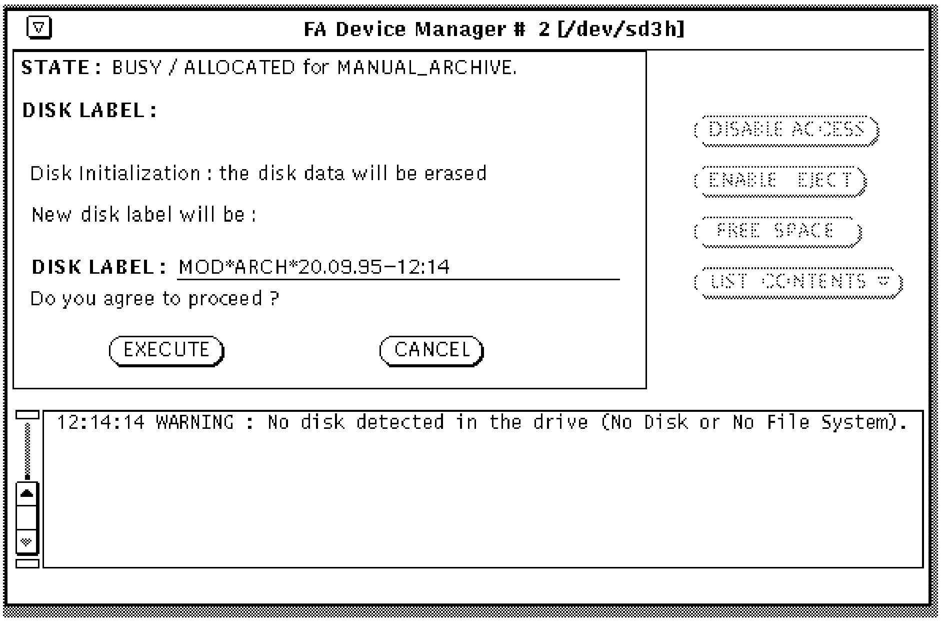
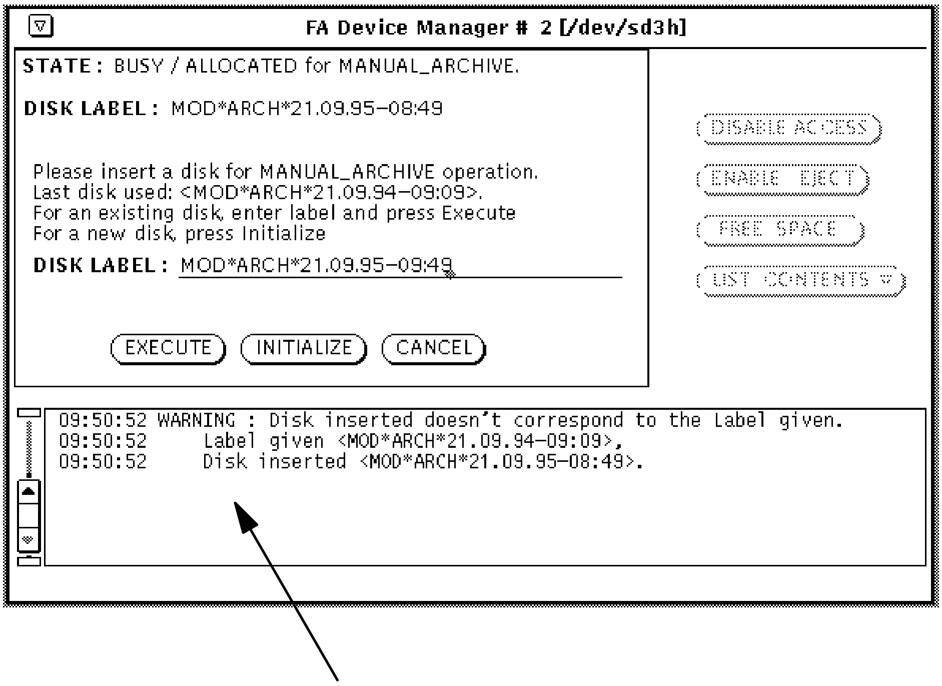


Figure 10–15 :*Disc Request for a Manual Archiving processing, Initialization confirmation.*

- For an old disk, already used for other Archiving/Retrieving operations, the operator must enter the disk label in place of the default one and click on the EXECUTE button. The last Archiving/Retrieving disk used is displayed to ease the operator work. At this stage, the FA SAS will check the disk label validity (See 10.2.4.1.2) and control that it is consistent with the disk inserted. Figure 10–16 shows the warning messages appearing in case of inconsistency between the disk label given and the actual disk inserted; the FA SAS returns automatically to the disk request display.



Warning message, inconsistency with label entered

Figure 10–16 :*Disc Request for Manual Archiving, Label given and disk inserted inconsistency.*

As an appropriate disk is inserted, the FA SAS can start the Archiving of the session. If the volume of the data to move from TRDB disk to the FA Medium is suffiscient, the FA SAS displays the progression of its work as shown in Figure 10–17.

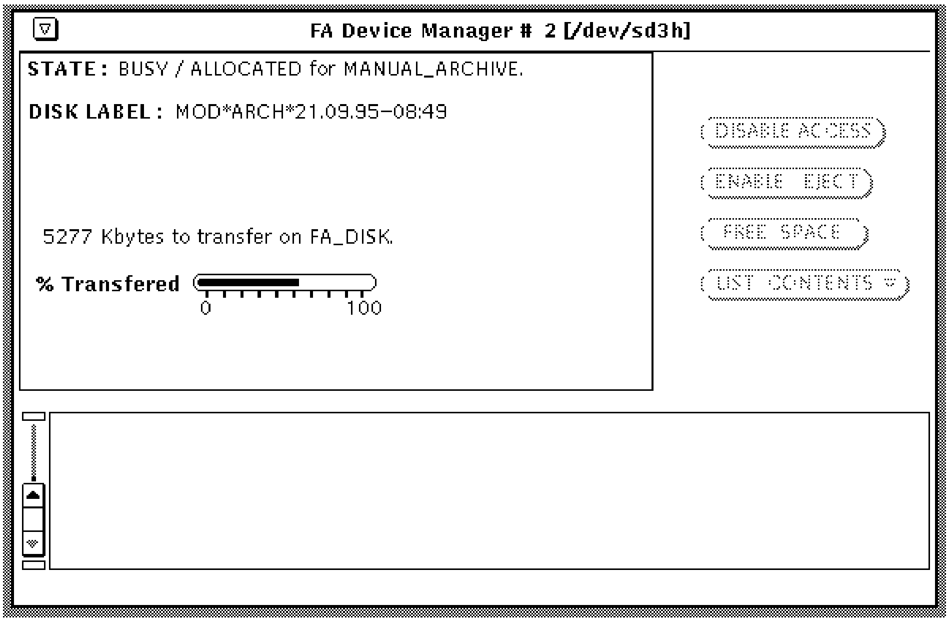


Figure 10–17 :*Manual Archiving processing, Progression Gauge.*

10.2.5.4.2 Retrieving of Data from the FA Medium

If a node issues a Retrieving command, the FA SAS will try to allocate an FA device according to the strategy defined in section 10.2.4.1.3. The FA SAS knows the disk containing the data to retrieve and requests it in the device allocated. If this disk is already inserted in an idle device, the FA SAS can execute its Retrieving of data immediatly. Otherwise, the FA SAS requests it to the operator as shown in Figure 10–18.

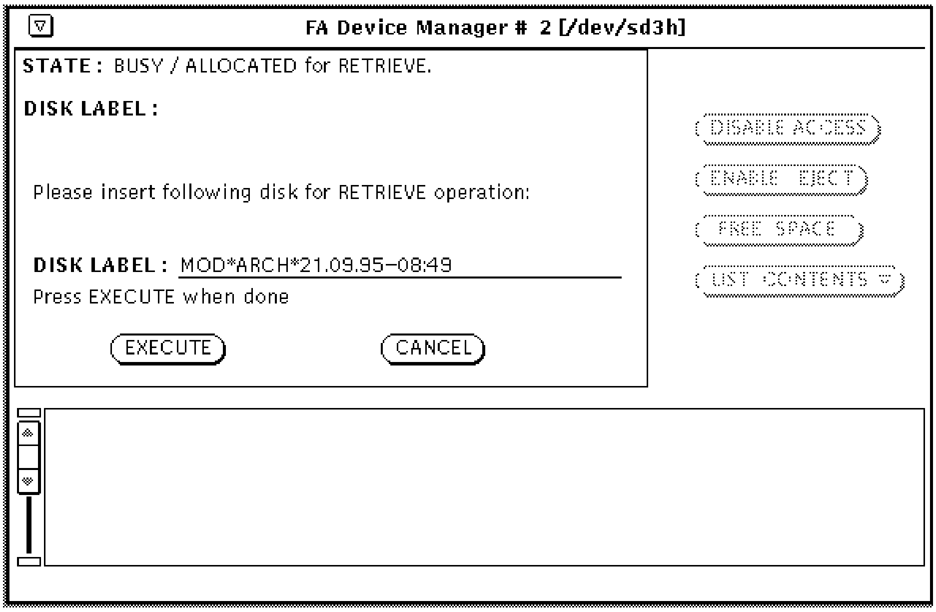
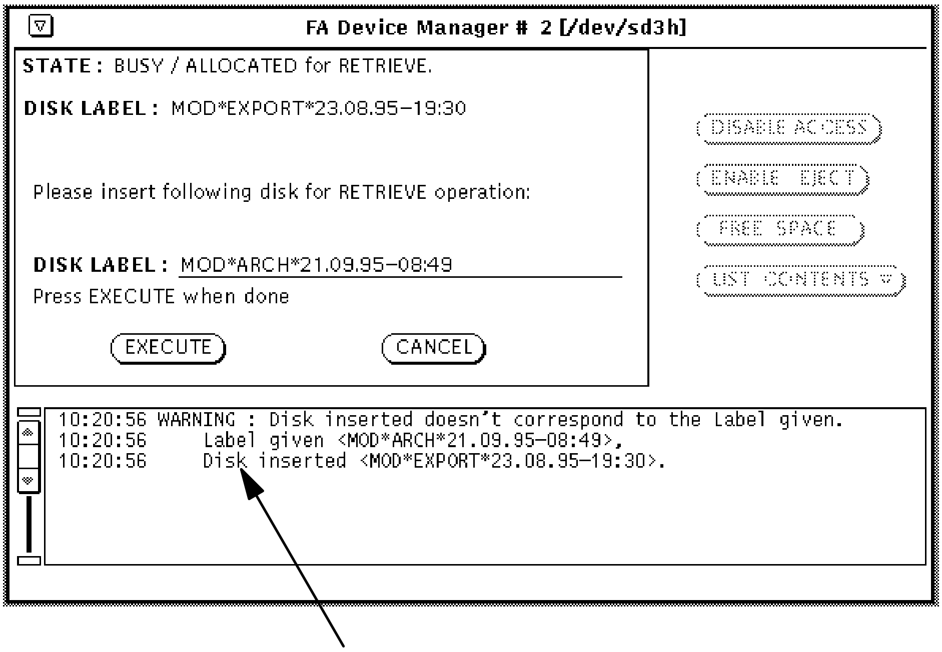


Figure 10–18 :*Disk request for a Retrieving of data.*

The device state is ”BUSY/ALLOCATED for Retrieve”, the buttons for administrative operations are disabled and the hardware ejection button is enabled to allow disk changes by the operator. The operator has to insert the requested disk (MOD*ARCH*21.09.95–08:49) and to click on the EXECUTE button.

If the operator inserts a wrong disk, a warning message is displayed in the error subwindow and the FA SAS returns automatically to the request disk display (Figure 10–19).



Warning message, inconsistency with label entered

Figure 10–19 :*Disk request for a Retrieving of data, label given and disk inserted inconsistency.*

As the appropriate disk is inserted, the FA SAS can start the Retrieving of data. If the volume of the data to copy from the FA Medium to the TRDB disk is sufficient, the FA SAS displays the progression of its work as shown in Figure 10–20.

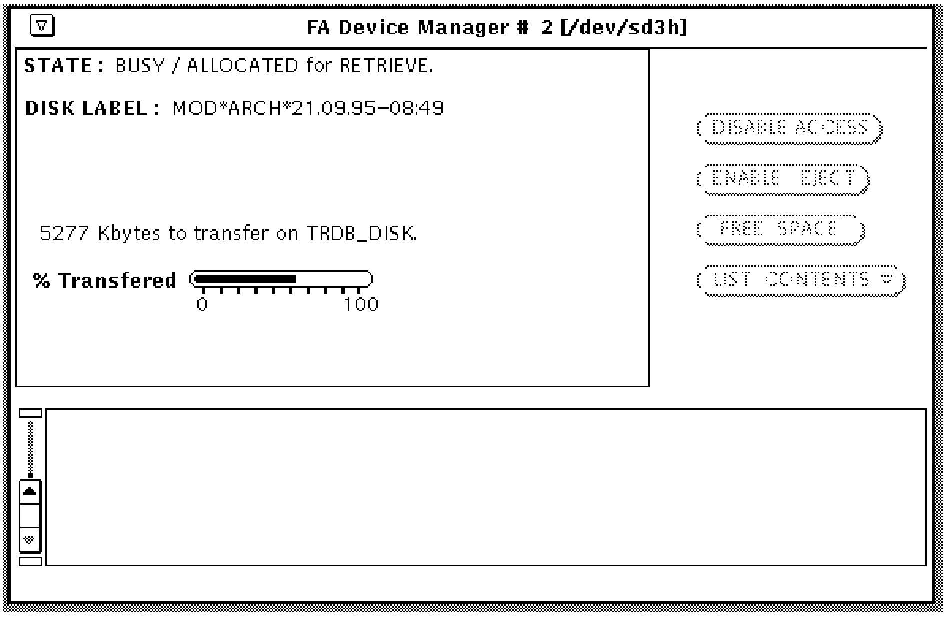


Figure 10–20 :*Retrieving of data, progression gauge display.*

10.2.5.4.3 Export of Session or Session Data to the FA Medium

If a node issues an Export command, the FA SAS will try to allocate an FA device according to the strategy defined in section 10.2.4.1.3.

If no Export/Import disk is available, the FA SAS will display a message requesting insertion of an appropriate disk, as shown in Figure 10–21. A time-out is associated to this disk request. Its value is a configurable parameter of the FA SAS program (See 10.2.7).

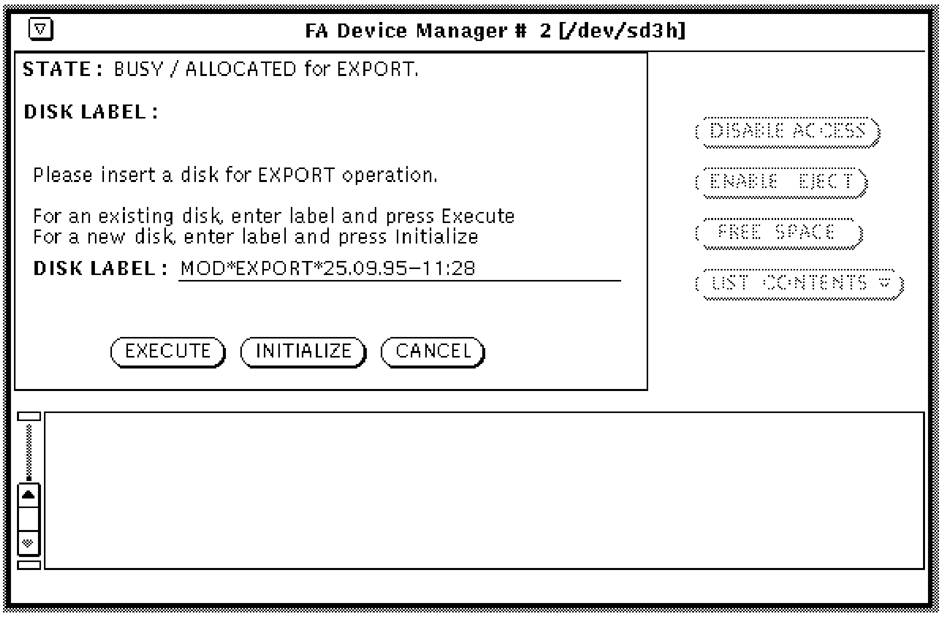


Figure 10–21 :*Disc Request for an Export processing.*

The device state is "BUSY/ALLOCATED for EXPORT", and the buttons for administrative operations are disabled. A default disk label is displayed and the hardware ejection button is enabled to allow disk changes by the operator.

The operator may want to use a new disk, an old EXPORT/IMPORT disk, or to re–initialize an old one. He can also cancel the operation, leading to a failure of the Export request.

- For a new disk, or to re–initialize an old one (remove data and create new label), the operator must click on the INITIALIZE button. The inserted disk will be initialized to the label proposed by the operator after confirmation (Figure 10–23).
  If the operator cancels the initialization at this stage, the FA SAS returns back to the disk request display. Note in the confirmation display (Figure 10–23) that the initialization label is not the default one shown in the Figure 10–22 but a disk label that the operator has previously entered. The FA SAS does not force the Export/Import disk label to particular values as it does for the Archiving/Retrieving disks.

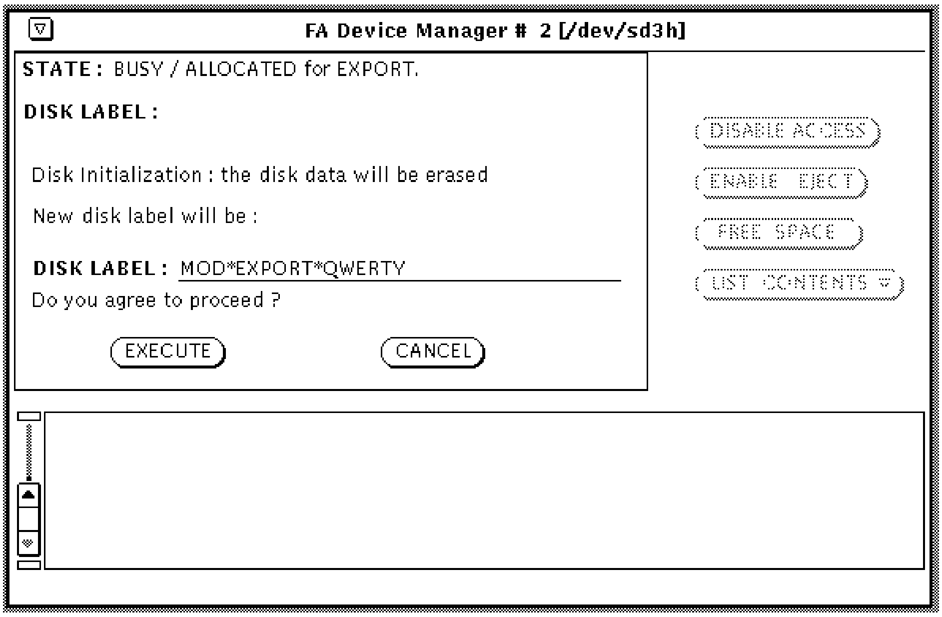


Figure 10–23 :*Disc Request for an Export processing, Initialization confirmation.*

- For an old disk, already used for other Archiving/Retrieving operations, the operator must enter the disk label in place of the default one and click on the EXECUTE button. At this stage, the FA SAS will check the disk label validity (See 10.2.4.1.2) and control that it is consistent with the disk inserted. Figure 10–24 shows the warning messages appearing in case of wrong disk label; Figure 10–25 shows the warning messages appearing in case of inconsistency between the disk label given and the actual disk inserted; the FA SAS returns automatically to the disk request display.

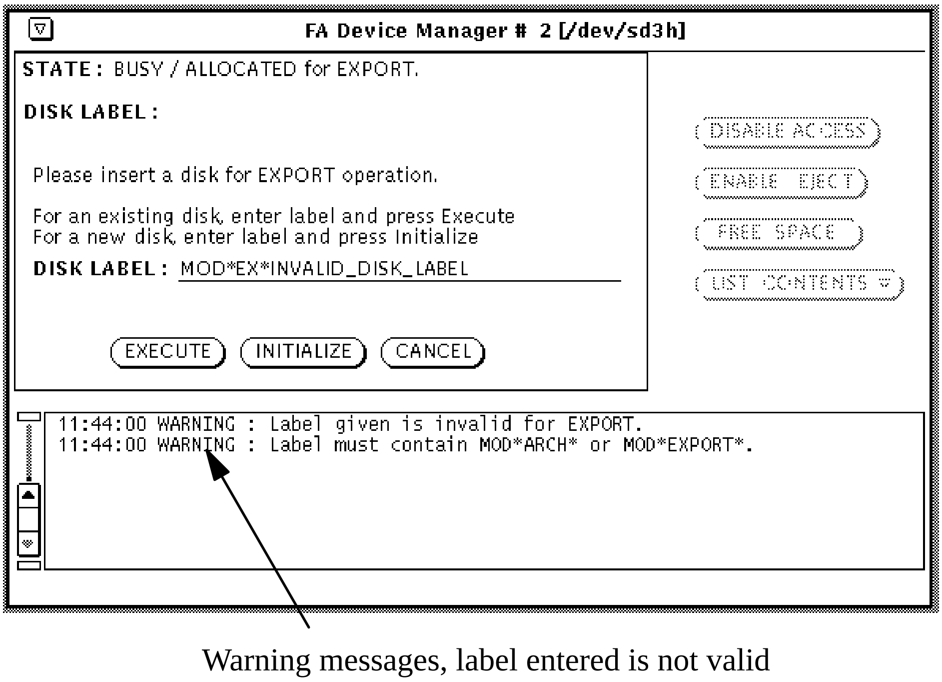


Figure 10–24 :*Disc Request for an Export processing, Label given is not valid.*

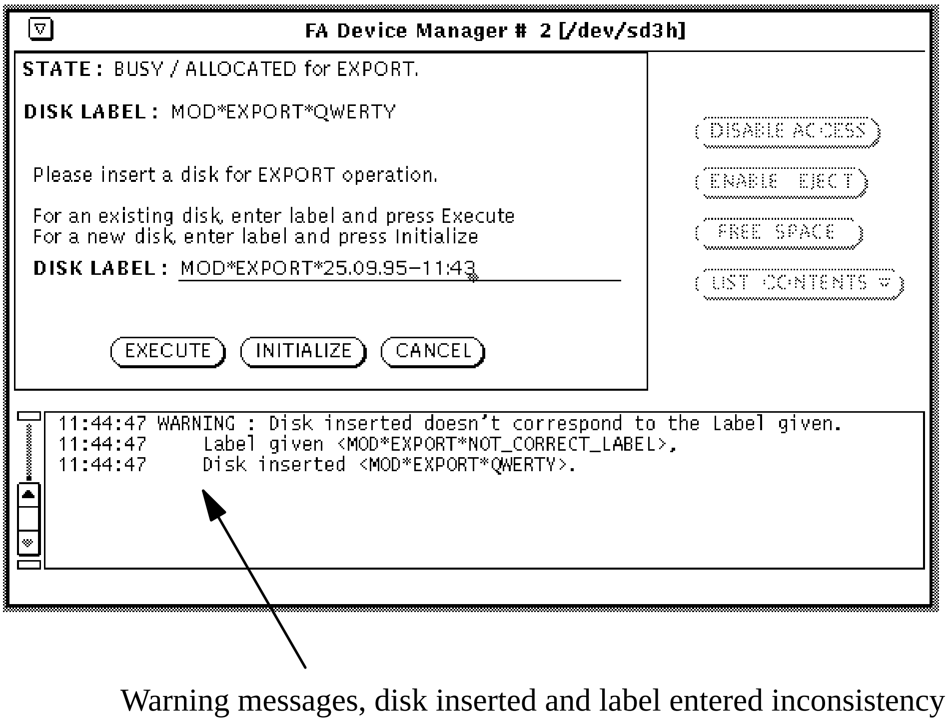


Figure 10–25 :*Disc Request for an Export processing, Label given and disk inserted inconsistency.*

As an appropriate disk is inserted, the FA SAS can start the Export of the session or session data. If the volume of the data to move from TRDB disk to the FA Medium is suffiscient, the FA SAS displays the progression of its work as shown in Figure 10–26.

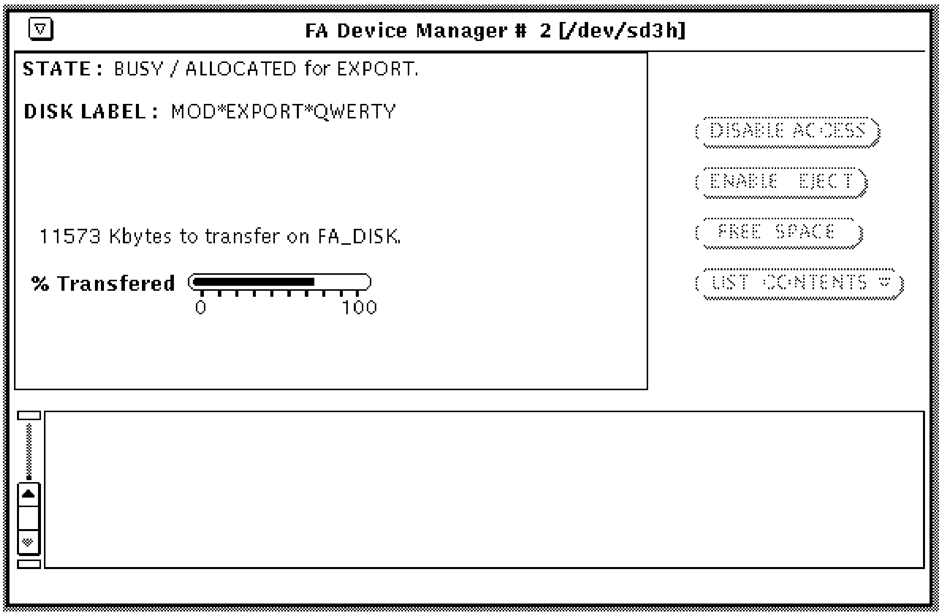


Figure 10–27 :*Export processing, Progression Gauge.*

10.2.5.4.4 Import of Session from the FA Medium

If a node issues an Import command, the FA SAS will try to allocate an FA device according to the strategy defined in section 10.2.4.1.3.

If no Export/Import disk is available, the FA SAS will display a message requesting insertion of an appropriate disk, as shown in Figure 10–28. The operator knows the label of the disk containing the data to import and has to enter it. DBS cannot propose a disk label as import disks were formatted at another site. A time-out is associated to this disk request. Its value is a configurable parameter of the FA SAS program (See 10.2.7).

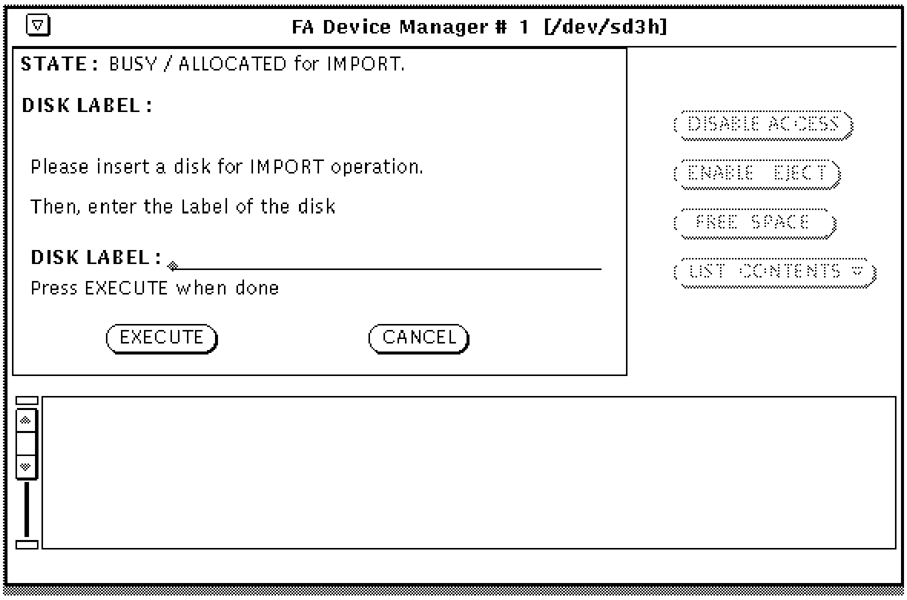


Figure 10–28 :*Disc Request for an Import processing.*

The device state is ”BUSY/ALLOCATED for IMPORT”, and the buttons for administrative operations are disabled. The hardware eject button is enabled to allow disk changes by the operator.

The operator has to enter a disk label and click on the EXECUTE button. At this stage, the FA SAS will check the disk label validity (See 10.2.4.1.2) and control that it is consistent with the disk inserted. Figure 10–29 shows the warning messages appearing in case of wrong disk label; Figure 10–30 shows the warning messages appearing in case of inconsistency between the disk label given and the actual disk inserted; the FA SAS returns automatically to the disk request display.

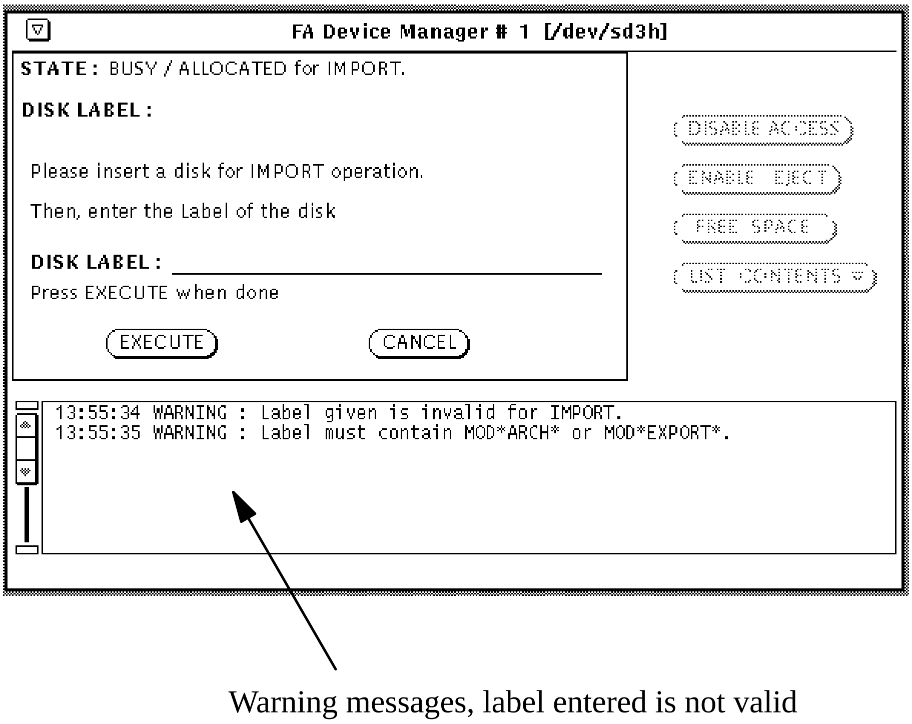


Figure 10–29 :*Disc Request for an Import processing, Label given is not valid.*

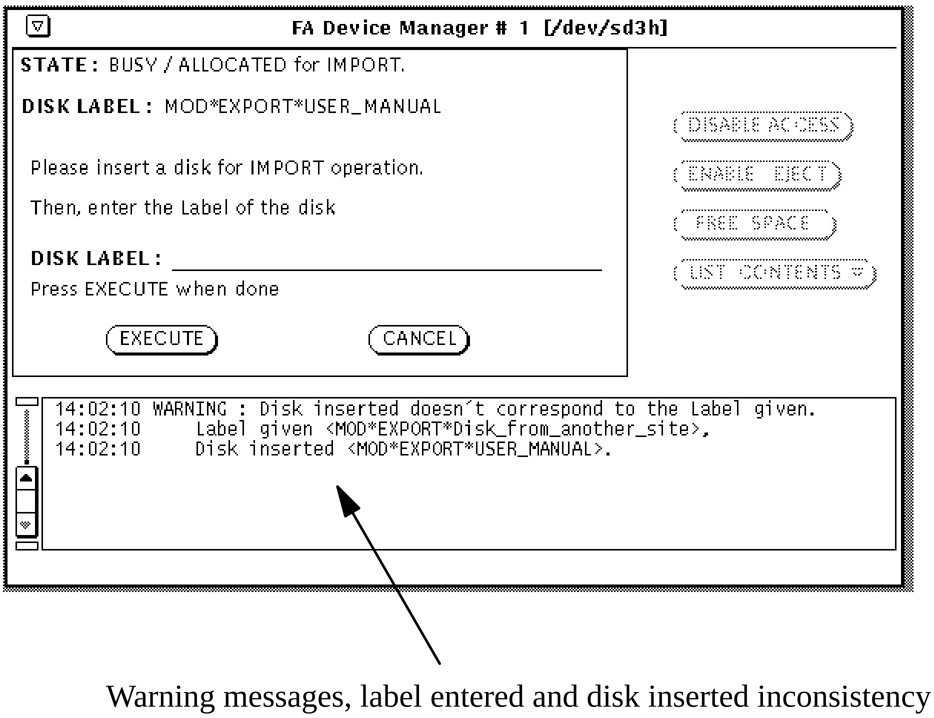


Figure 10–30 :*Disc Request for an Import processing, Label given and disk inserted inconsistency.*

As the appropriate disk is inserted, the FA SAS can start the Import of the session. If the volume of the data to move from TRDB disk to the FA Medium is suffiscient, the FA SAS displays the progression of its work as shown in Figure 10–31.

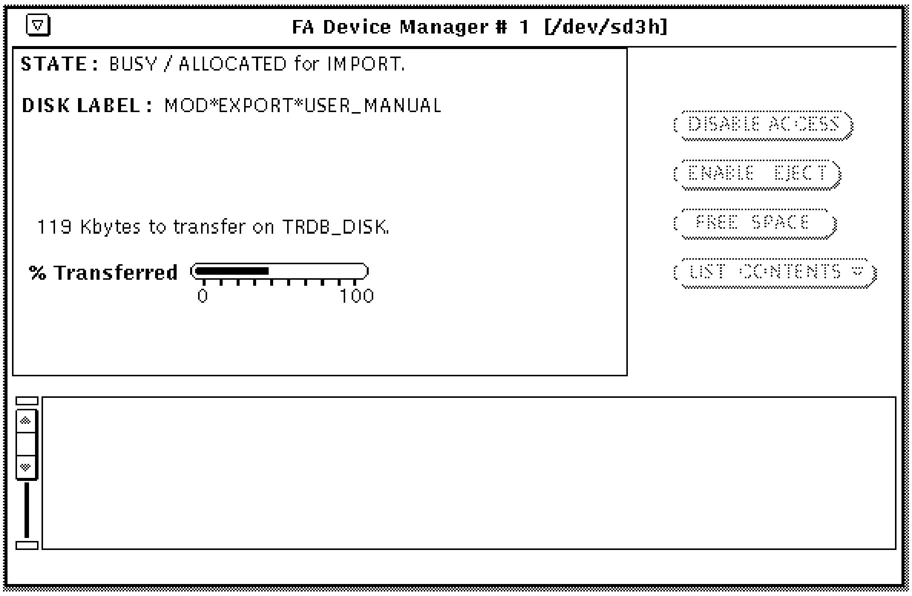


Figure 10–31 :*Import processing, Progression Gauge.*

10.2.5.4.5 Automatic Archiving of Execution Sessions to FA Medium

An application can initialize a session with the Automatic Archiving option. It means that the session data will be automatically archived to the FA medium when the TRDB disk space reaches a predefined level; DBS does not wait for the session to be closed by the application. In this case, the FA SAS is requested to allocate an FA device for the whole session duration; i.e. the device allocated will be immediately available for Automatic Archiving of data and the operator cannot access it. The device is freed at the session closure by a DBS request.

The first step of the Automatic Archiving processing is to allocate a device with an appropriate disk inside. If no Archiving/Retrieving disk is available, the **FA SAS will display a message requesting insertion** of an appropriate disk. A time-out is associated to this disk request. Its value is a configurable parameter of the FA SAS program (See chapter 10.2.7). The disk request handling is similar to the one described in the previous sections.

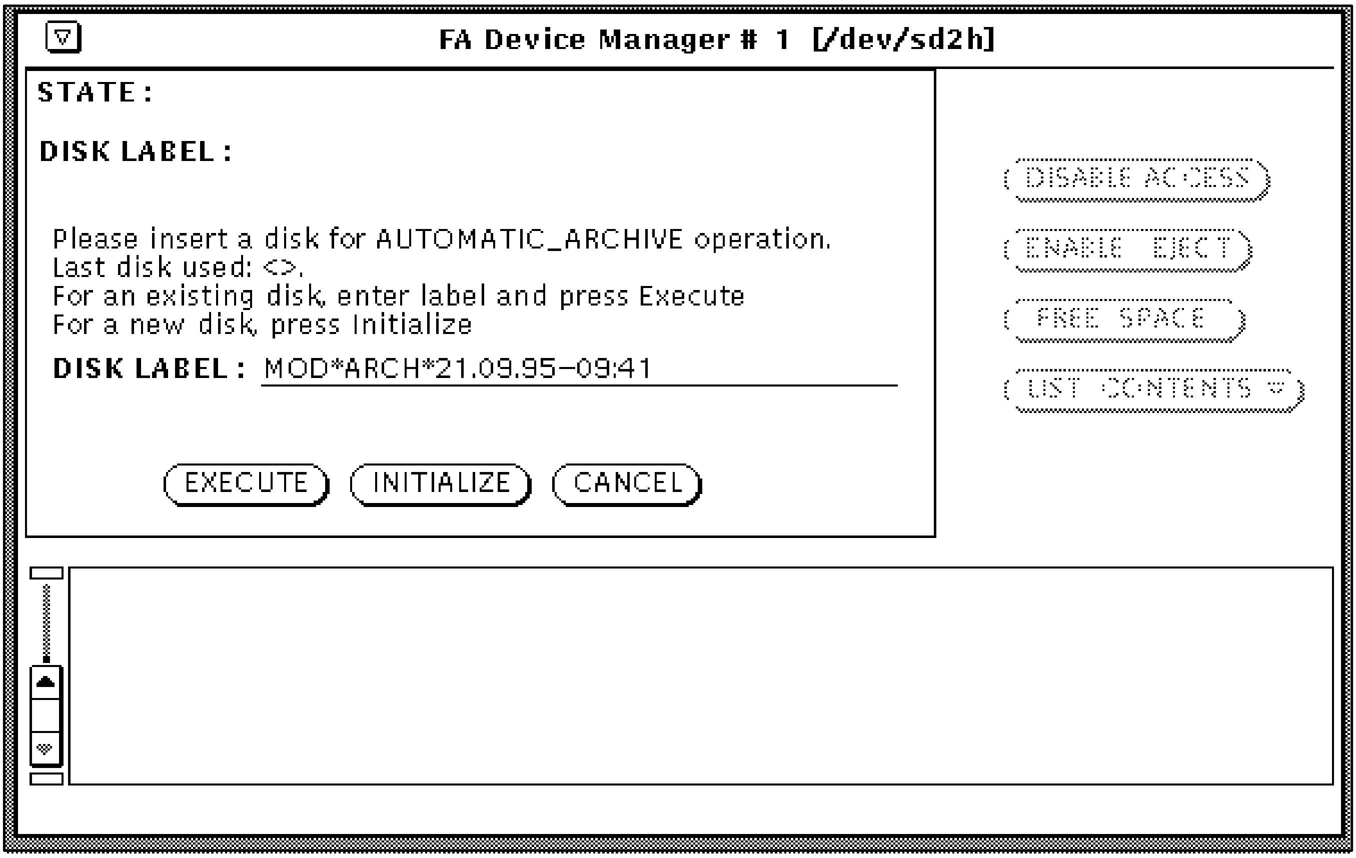


Figure 10–32 :*Disc Request for an Automatic Archiving processing*

As soon as an appropriate disk is inserted, the device is allocated for Automatic Archiving, waiting for DBS requests to archive session data.

If the free space on the FA disk becomes too small to continue the Automatic Archiving processing, the FA SAS will request another disk from the operator. Such a disk request is handled the same way as the initial one.

After each archiving of session data, the device control window displays the state ’IDLE/ALLOCATED for AUTOMATIC ARCHIVING’ and the current free space on the disk (See figure 9–75). This allows the operator to follow the filling of the current disk and to prepare a new one if required.

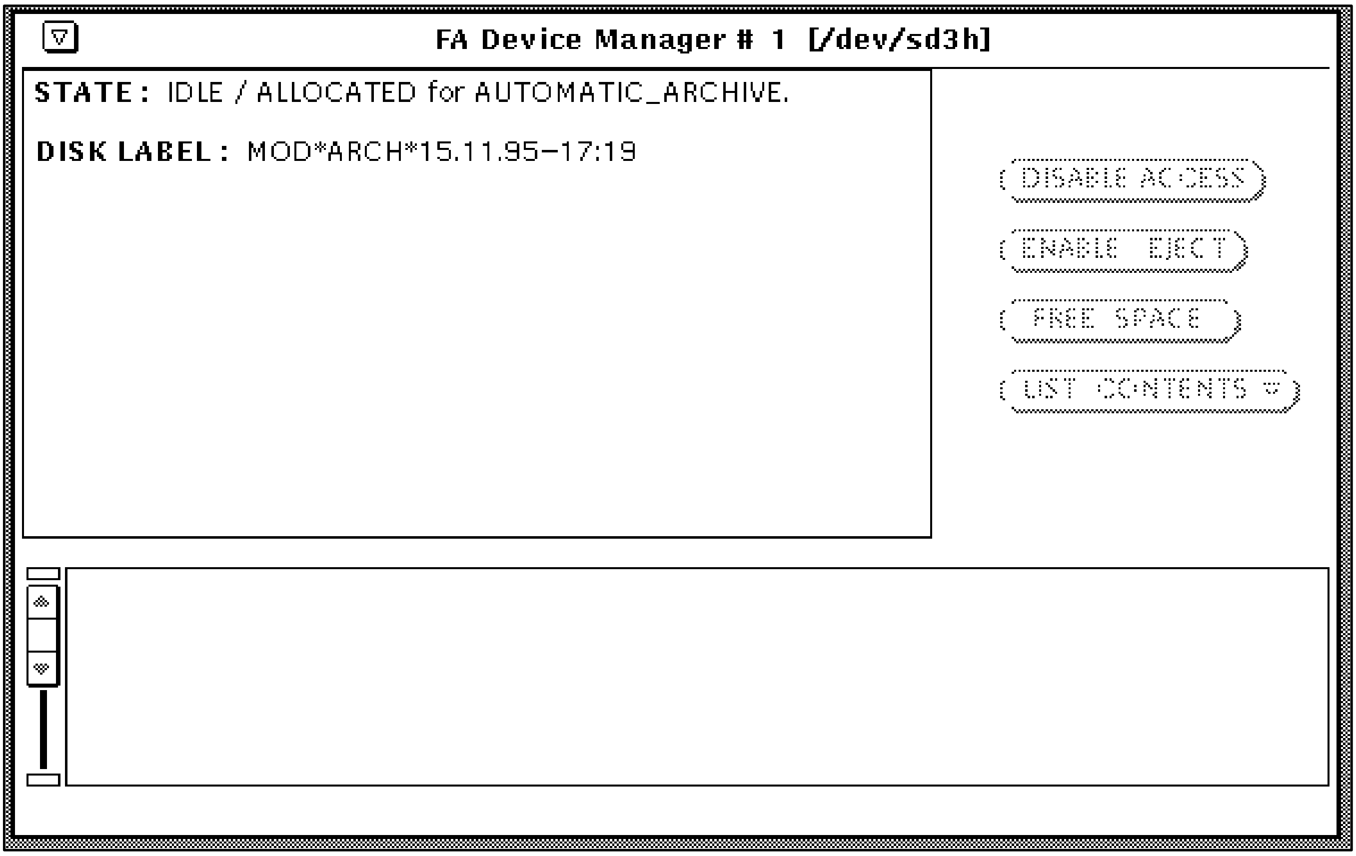


Figure 10–33 :*Free Space left during Automatic Archiving Processing*

10.2.5.5 Error management

The FA SAS reports errors and warnings using the CGSI error report facilities and/or the message subwindows of its MMI.

The warnings and information that concern only the FA SAS operator are displayed only in the message subwindows (e.g. disk inserted does not correspond to the label given by the operator, no disk detected in the drive, operator requested disabling of the DBS access...). Figure 10–34 shows such a warning addressed to the operator.

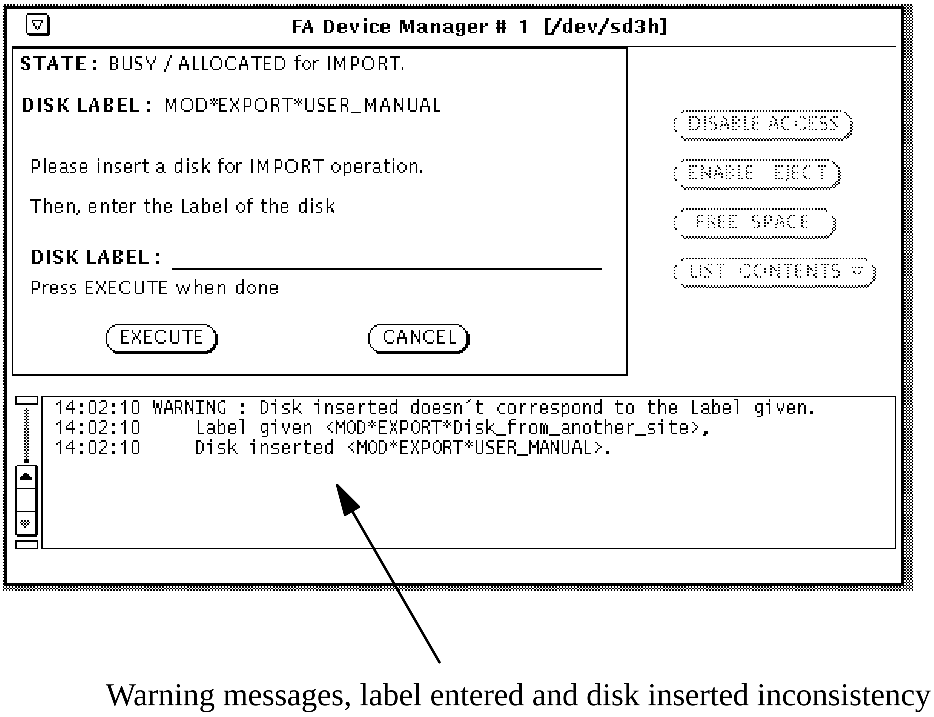
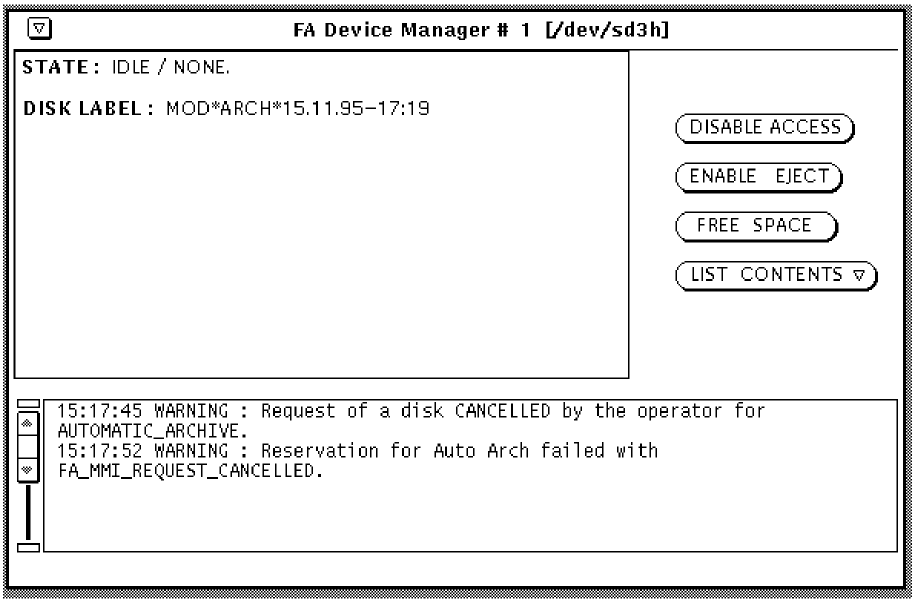


Figure 10–34 :*Disc Request for an Import processing, Label given and disk inserted inconsistency.*

The FA SAS reports software or handling errors/warnings that affect other applications using the CGSI error report facilities and the message subwindows of its MMI (e.g. DBS disk request cancelled or not answered by the operator, problem accessing the optical disk...). The Figure 10–35 shows such a message addressed to the operator and to the CGSI.

- **Information addressed to the operator**:



- **Information addressed to the CGSI**:

  timestamp : 28/08/95–15:17:45:06
  extra_text :
  error_spec mnemonic : FA_SAS_WARNING
  error_spec error severity : ADVISORY
  error_spec error text : *** WARNING from DISC_ACCESS.MOUNT_DISC
  error_spec error suppl text : ***WARNING : Request of a disk CANCELLED by the operator for AUTOMATIC_ARCHIVE.***
  error_spec ack flag : ACK_NOT_REQUIRED
  handling : LOG_AND_DISPLAY
  tags test node : ***FA–SAS***
  tags group name : TRDB
  user : *
  node : *

Figure 10–35 :*Example of warning reported to the operator and to the CGSI.*

All the messages displayed in the message subwindows of the FA SAS MMI are also written in the following files :

- ”**$GSAF_HOME/dbs/data/FA_SAS_1.log**” for the first FA device,
- ”**$GSAF_HOME/dbs/data/FA_SAS_2.log**” for the second one.

Most of the error messages used by the FA SAS are configurable. They are defined by the FA SAS configuration file (See 10.2.7).

10.2.6 Preparation of FA SAS Disks

First, the disk type must be defined in the file ***/etc/format.dat*** to ease further formatting of magneto–optical disks. The lines to insert as super–user were in our example:

#
 # Sony Magneto Optical Disk
#
*disk_type = ”SONY–SM0–F521–00–1.04” *
*: ctlr = SCSI : ncyl = 37523 : acyl = 2 : pcyl = 37525 *
: nhead = 1 : nsect = 31 : rpm = 3000 : bpt = 3600
#
 # Sony Magneto Optical Disk
#
*partition = ”datadisk” *
*: disk = ”SONY–SM0–F521–00–1.04” : ctlr = SCSI *
: c = 0, 1163213 : h = 0, 1163213

There is commonly no need to format a new disk, it is already formatted by the supplier. The super–user however must create a file system on any magneto–optical disk before providing it to FA SAS. After creation, it has to be made sure that the root filesystem is owned by the CGS administrator. This can be performed by running the UNIX chown command:

EXAMPLE:
mount /dev/dsk/<*devive_name*> /mnt
chown cgsadmin:cgs /mnt
umount /mnt

In the following, we give an example of the commands to issue, the menus displayed could be different for another configuration:

1) creating a new file system on the formatted disk:

Operator Action	Output on the Screen
type ’**newfs /dev**/*rsd3h’* or, if a lot of small files (a few Kbytes) are expected : type ’**newfs –i 16384 –t 32 –c 32 –m 3 /dev**/*rsd3h’* *<Return>*	
	setting optimization for space with minfree less than 10 Warning: 404 sector(s) in last cylinder unallocated /dev/rsd3h: 1163212 sectors in 1173 cylinders of 32 tracks, 31 sectors 595.6MB in 37 cyl groups (32 c/g, 16.25MB/g, 960 i/g) super–block backups (for fsck –b #) at: 32, 31808, 63584, 95360, 127136, 158912, 190688, 222464, 254240, 286016, 317792, 349568, 381344, 413120, 444896, 476672, 508448, 540224, 572000, 603776, 635552, 667328, 699104, 730880, 762656, 794432, 826208, 857984, 889760, 921536, 953312, 985088, 1015840, 1047616, 1079392, 1111168,1142944,

Then the disk is ready for FA SAS use.

10.2.7 Configuration file of FA SAS

```
! ————————————————————
! FA SAS configuration file
! ————————————————————
!
! SCCS version identifier, do not change
! Version: %Z%%Z%
!
! This file is read at elaboration time by the FA SAS process
! The pathname of this file : $GSAF_HOME/dbs/config/fa_sas_configuration_file.def
!
! The syntax of each line must follow one of the following templates :
!
!    1) <TEXT_STRING> <VALUE_TYPE> <VALUE>
!                     Define a new parameter
!
!              TEXT_STRING must not contain any blank character
!                     VALUE_TYPE must be :     1 –> INTEGER,
!                                                      3 –> STRING
!              if VALUE_TYPE is a string then
!              <VALUE> is can be :
!                                     i) a string composed with blank characters but
!                                           enclosed into delimiters ””.
!                                        Example : ”This is a valid string with blank characters”
!                                    ii) a single word with no blank character
!                                        Example : THIS_IS_A_VALID_WORD
!
!   2) !<TEXT_STRING>
!                     Identify a comment line. First character must be ”!”
!
!   3) !#
!                     Indicates end of configuration file
!
!
!  ————————————————————————————————————————
!      NB:   i) Parameter names in this file must not be changed
!           ii) Parameters must not be deleted even if they are not used
!  ————————————————————————————————————————
!
! =========
!  FA_SAS
! =========
!
! The FA SAS is able to handle ”virtual” MO devices, i.e. a directory located
! in the filesystem to be treated as MO device. A virtual device will replace
! a real device, i.e. only two devices can be handled by the FA SAS as a maximum.
! Prerequisite: the device must be block–structured or a device provided by
! remote machines.
```

```
! Examples of a definition:
!   block structured device
!   FA_DEVICE_FILENAME1 3 "/dev/dsk/c0t0d0s7"
!   device provide by remote machine
!   FA_DEVICE_FILENAME1 3 "cgs–test:/cgs/users/cgsadmin/virtual_MO"
!
! Please note that it is easily possible to allocate a virtual device on the
! same machine (database server) by defining it as remote.
! A definition like
!  FA_DEVICE_FILENAME1 3 "/cgs/users/cgsadmin/virtual_MO"
! will cause an error on UNIX level (Message: "not a block device").
!
! The device file names will be used as link to the "real" MO–device
! (example: /dev/mo1 is a symbolic link to /dev/dsk/c0t3d0s2)
! Note 2 device names must always be provided
!      ————————————————————————————————
FA_DEVICE_FILENAME1 3 "/dev/mo1"
FA_DEVICE_FILENAME2 3 "/dev/mo2"
!
!
! The following value denotes the device which will be selected for
! automatic archiving in batchmode, i.e. without user interrogation
DEFAULT_DEVICE_FOR_AUTOMATIC_ARCHIVING 1 1
!
!
! The Name of the DBS Owner and privileged user:
! ————————————————————————————————————————
! The files on the TRDB disk and on the FA disk are owned by 'DBS OWNER'.
!
!  Please Note the following:
!  During "install_dbs" the ownername "DBS OWNER NAME DEFAULT" will be substituted
!  by the onsite DBS OWNER UNIX username. See $DBS_HOME/util/common/install_dbs.
!   NB: 'DBS OWNER' must be identical to 'DBS OWNER NAME DEFAULT' in dbs_configuration_file.def
!
DBS_OWNER_NAME 3 "DBS_OWNER_NAME_DEFAULT"
!
! The Name of the environment variable defining the FA
! SAS Printer (used by MMI operation Print Detailed List).
! ————————————————————————————————————————————————
FA_SAS_PRINTER 3 "FA_SAS_PRINTER"
!
!
! The Name of the files containing the messages recorded
! from the Text Subwindows in the FA MMI .
! ————————————————————————————————————————————
! (Log files can be found in $VICOS_CEN_DBS_HOME)
FA_SAS_1.log 3 "FA_SAS_1.log"
```

```
FA_SAS_2.log 3 ”FA_SAS_2.log”
!
!
! FA MMI request Time Out
! If the operator does not reply in FA_MMI_TIME_OUT seconds (eg for a mount disk request)
! status FA_MMI_TIME_OUT will be returned to the caller
! ————————————————————————————————————
FA_MMI_TIME_OUT 1 120
!
! The information needed for error reporting through CGSI:
! ————————————————————————————————————
! Definition of the FA SAS tag fields for error reporting :
ERROR_GROUP_NAME 3 ”TRDB”
ERROR_NODE_NAME 3 ”FA–SAS”
!
! Display showing file transfer progress :
! ——————————————————————————
! A diagram showing the progress pf a FA job will be displayed if the file size (kbytes)
! is greater than PROGRESSION_TASK_THRESHOLD.
! This diagram will be updated every PROGRESSION_TASK_DELAY
!
PROGRESSION_TASK_DELAY 1 1
PROGRESSION_TASK_THRESHOLD 1 0
!
!
! Error messages :
! ———————————
! package MMI_DISPLAY:
!          ————————
! Following error/warning messages may be displayed by the package MMI_DISPLAY:
!              ——————————————————————————
!              !  Note DO NOT CHANGE THESE MESSAGES  !
!              ——————————————————————————
!  Request of a disk interrupted by operator Time out:
REQ_DISK_TIMEOUT_MSG 3 ”WARNING : A timeout occured during disk request dialog. ”
!  Request of a disk cancelled by the operator:
REQ_DISK_CANCEL_MSG 3 ”WARNING : Request of a disk CANCELLED by the operator for ”
!  Ejection of a disk interrupted by operator Time out:
EJECT_TIMEOUT_MSG 3 ”WARNING : Ejection of a disk interrupted by operator TIME OUT. ”
!  Trial to initialise FA MMI windows after they have been already stopped:
WINDOW_LOOP_STOPPED_MSG 3 ”WARNING : Request to start FA MMI after it has been stopped.”
!
! package DISC_ACCESS:   error messages appearing in the text panel
!          ————————   of the FA MMI and/or reported to the CGSI.
!  Disk not correctly mounted by the mount operation:
NO_DISC_MOUNTED 3 ”WARNING : The disk is not correctly mounted.”
!  Read only disk has been mounted:
READ_ONLY_MSG 3 ”WARNING : Read only disk has been mounted.”
```

! No disk detected in the drive by the mount operation:
NO_DISC_ERROR 3 ”WARNING : No disk detected in the drive (No Disk or No File System).”
! Initialisation of the disk not valid for this type of operation:
EMPTY_DISC_MSG 3 ”WARNING : The mounted disk is invalid for this kind of operation (”
! Formatting cancelled by the operator:
DISC_NOT_ERASED_MSG 3 ”WARNING : Operator Cancel; The mounted disk has not been formatted.”
! Disk inserted is not Formatted for operation:
DISC_WRONG_LABEL_MSG 3 ”WARNING : Disk inserted is not Formatted for ”
! Disk inserted doesn’t correspond to the Label given:
LABEL_AND_DISC_INCONSISTENCY_MSG 3 ”WARNING : Disk inserted doesn’t correspond to the Label given.”
! Invalid disk for operation:
INVALID_DISK_MSG 3 ”WARNING : Invalid disk for ”
! Label given is invalid for operation:
LABEL_WRONG_MSG 3 ”WARNING : Label given is invalid for ”
! Writable disk is mandatory:
RO_DISC_MSG 3 ”WARNING : Writable disk is mandatory for the required operation.”
! Not enough free space on FA disk:
FS_FULL_MSG 3 ”WARNING : Not enough free space on FA disk.”
! Not enough free space on TRDB disk:
TRDB_FS_FULL_MSG 3 ”WARNING : Not enough free space on TRDB disk to copy from FA.”
! The environment variable giving the FA SAS Printer is not defined:
PRINTER_NOT_DEF_MSG 3 ”WARNING : The environment variable giving the FA SAS Printer is not defined.”
!
! package DEVICE_CONTROLLER:
! ——————————
! Following error/warning messages may be displayed by the DEVICE_CONTROLLER
! ————————————————————————
! ! Note DO NOT CHANGE THESE MESSAGES !
! ————————————————————————
! The device is requested to stop activity.
STOP_WARNING 3 ”WARNING : Device requested to stop activity.”
! The device is requested to disable DBS access.
DISABLE_WARNING 3 ”WARNING : Device requested to disable DBS access.”
! The device is requested to enable DBS access.
ENABLE_WARNING 3 ”WARNING : Device requested to enable DBS access.”
!
!
! package FA_SAS_MUX:
! ————————
! Following error/warning messages may be displayed by the DEVICE_CONTROLLER
! ————————————————————————
! ! Note DO NOT CHANGE THESE MESSAGES !
! ————————————————————————
! The initialisation of the MMI windows failed with ’STATUS’.
MMI_INIT_FAILURE_MSG 3 ”Xview Interface Init failed with ”
MUX_INIT_FAILURE_FOL_MSG 3 ”; Final Archive Init Aborted.”

```
! The initialisation of the Device Controllers failed with 'STATUS'.
DEV_INIT_FAILURE_MSG 3 "Device Controller Init failed with "
! The initialisation of the FA SAS Comms failed with 'STATUS'.
COMMS_INIT_FAILURE_MSG 3 "FA SAS Comms Init failed with "
!
!
!
! Tasks Stack size:
! ––––––––––––––––––
! Progression display tasks (in mode DEBUG, messages printed in the MMI)
! Stack size in Bytes.
! Note  '_' used to enhance readability eg 32_000 => 32 kbytes
!
PROGR_DISPL_TASK_SIZE 1 256_000
! Device Controller tasks
DEV_CTRL_TASK_SIZE 1 256_000
! Window loop tasks
WO_LOOP_TASK_SIZE 1 256_000
! FA SAS Mux main task
FA_SAS_MUX_STACK_SIZE 1 256_000
!
! Minimum percentage of free space in the FA partition needed to work.
! ––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
PERCENT_OF_FREE_SPACE 1 1
!
!#
```

10.3 The Recovery Scripts

The Recovery Scripts allows the user to recover some inconsistencies appearing in the TRDB, due to DBS operation failures.

10.3.1 Getting started

The Recovery Scripts are launched by invoking the C shell script *$DBS_HOME/util/sun5/recovery/recovery.csh* in the *DBS owner*'s user account.

The following menu is displayed :

```
– DBS RECOVERY SCRIPTS MAIN MENU –
___________________________________

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_ERR_xxx)

0. Exit

  Enter your choice :
```

Figure 10–36 :*The Main Menu of the Recovery Scripts*

10.3.1.1 Execution Session

Select the Execution Session Menu by entering '1' in the Main Menu. The Execution Session Menu allows to call all the recovery scripts operations related to test execution sessions. It is shown in Figure 10–37. The

user has to chose between 3 items. The choice 0 returns to the main menu.

```
– DBS RECOVERY SCRIPTS MAIN MENU –

      1. Execution Session
      2. Evaluation Session
      3. DBS Error Number (DBS_ERR_xxx)

      0. Exit

 Enter your choice :  1

   – 1. Execution Session Menu –

      1. Session Data Diagnostic
      2. Delete Session
      3. Session is Used
      4. File Storage Failure
      5. Close Session
      6. List Sessions
      7. File Transfer from Local Dir Failures
      8. Close aborted test sessions to NCL

      0. Exit

   Enter your choice : 1

      1.1 Session Data Diagnostic

      Enter session name :
```

Figure 10–37 :*Selection of the Execution Session Menu from the Main Menu*

10.3.1.1.1 Execution Session Diagnostic

As shown in the figure Figure 10–37, if the selection in the Execution Session Menu is ’1’, the user is asked for the session whose data diagnostic is required.

This functionality allows the scanning of the session data references and the display of a report. The information provided summarizes the session status, and helps the user to decide which recovery action is required. The Figure 10–38 shows such a report for a session named EXEC_BEN_005 which is partially retrieved from the Final Archive.

A report contains 8 sections. The 5 first sections list the references of the session data files. The division is based on their FILE_STATUS :

- Central ACcessible (CAC) files: these are files normally stored into the TRDB, accessible for evaluation ('on–line').
- Local ACcessible (LAC) files: these are files that local applications expect to be stored into the TRDB by Central DBS; they are not yet accessible for evaluation. This state is normally temporary during the storage into the TRDB by Central DBS (OPEN sessions). A file whose storage into the TRDB failed may remain in the state Local Accessible
- Central Not Accessible (CNA) files: these are files managed by Central DBS but not accessible for evaluation. It can be an error case.
- Local Not Accessible (LNA) files: these are files not accessible for evaluation and to Central DBS. It is an error case.
- ARChived (ARC) files: these are files archived on an Final Archive medium and not retrieved on the TRDB disk (not on–line).

The sixth section of a report contains a summary of the session Oracle Event table and its reference. If the Events of a session are on–line, an Oracle Event table must exist.

The seventh section lists the contents of the session WORK directory. This directory contains only temporary files. As soon as the session is closed, it should be empty. It may contain file whose storage into the TRDB failed.

The last section displays the session state. It checks the consistency of the session data state compared with the session state and summarize the situation.

The last report generated by the Recovery Scripts is maintained into the file ***$DBS_HOME/util/sun5/recovery/work/Recovery.diagnostic***.

Recovery Execution Session Diagnostic : *<EXEC_BEN_005>*

The diagnostic analyses the TRDB references for session EXEC_BEN_005
i.e. the filename location, the session state/session data state.
Files are classified as :

– Central Accessible (normally stored into the TRDB, accessible for evaluation),
– Local Accessible (files that Local nodes expect to store into TRDB),
– Central Not Accessible (files that are not accessible for evaluation),
– Local Not Accessible (files that are not accessible for evaluation and to Central),
– Archived (files that are archived on an FA SAS Medium).

The latest diagnostic is kept into a file : ***$DBS_HOME/util/sun5/recovery/work/Recovery.diagnostic***

1.1 Listing of Central Accessible files referenced in the TRDB

Reference of File found : $DBS_HOME/data/EXECUTION/EXEC_BEN_005/RAW_DATA/TEV_06/qwerty23
Reference of File found : $DBS_HOME/data/EXECUTION/EXEC_BEN_005/RESULT/qwerty233
Reference of File found : $DBS_HOME/data/EXECUTION/EXEC_BEN_005/EVENT/EVENT_TABLE04–03–96_09:31:28:616
It is a reference for an event file, file not found but event table must exist (see 1.6)
Listing of Central Accessible files finished :3 file(s)

1.2 Listing of Local Accessible files referenced in the TRDB

No Local Accessible file referenced in TRDB

Listing of Local Accessible files finished :0 file(s)

1.3 Listing of Central NOT Accessible files referenced in the TRDB

No Central NOT Accessible file referenced in TRDB
Listing of NOT Accessible files finished :0 file(s)

1.4 Listing of Local NOT Accessible files referenced in the TRDB

No Local NOT Accessible file referenced in TRDB
Listing of NOT Local Accessible files finished :0 file(s)

1.5 Listing of Archived files referenced in the TRDB

Reference of File found : Final_Archive/EXECUTION/EXEC_BEN_005/ENG_VAL/TEV_06/TEV_06_04–03–96_09:29:22:558.EVL
Disk name is : MOD*ARCH*15.02.96–17:35
Listing of Archived files finished :1 file(s)

Reference of Retrieved File found : $DBS_HOME/data/EXECUTION/EXEC_BEN_005/RAW_DATA/TEV_06/qwerty23
(should match a Central Accessible Reference).
Disk name is : MOD*ARCH*15.02.96–17:35
Reference of Retrieved File found : $DBS_HOME/data/EXECUTION/EXEC_BEN_005/RESULT/qwerty233
(should match a Central Accessible Reference).
Disk name is : MOD*ARCH*15.02.96–17:35
Reference of Retrieved File found : $DBS_HOME/data/EXECUTION/EXEC_BEN_005/EVENT/EVENT_TABLE04–03–96_09:31:28:616
(should match a Central Accessible Reference).
Disk name is : MOD*ARCH*15.02.96–17:35
Listing of Retrieved files finished :3 file(s)

1.6 Listing of Event Oracle Table referenced in the TRDB

TRDB does contain an Oracle Event Table for EXEC_BEN_005
Event Number : 1
Listing of Event Oracle Table finished

1.7 Listing of Session Work Directory

Session Work Directory does exist
path : $DBS_HOME/data/WORK/EXEC_BEN_005
content : Directory is empty
Listing of Session Work Directory finished

1.8 Checking of Session data

Session is recorded as ARCHIVED on FA Medium

Data could be on FA Medium only (archived) or
on FA Medium and on TRDB disk (retrieved from FA).
1 file(s) is(are) referenced as on FA Medium only
3 file(s) is(are) referenced as Central Accessible and Retrieved
Checking of Session data finished

Figure 10–38 :*Execution Session Data diagnostic report.*

10.3.1.1.2 Delete Execution Session Menu

As shown in the Figure 10–39, if the selection in the Execution Session Menu is ’2’, the Delete Execution Session Menu is displayed.

– 1. Execution Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Close Session
6. List Sessions
7. File Transfer from Local Dir Failures
8. Close aborted test sessions to NCL

0. Exit

Enter your choice : 2

1.2 Delete Session

– 1.2 Delete Execution Session Menu –

1. Delete Default Test Session Data (Session Refs remain)
2. Delete the On–Line Data of an Archived Test Execution Session (Refs remain)
3. Delete Completely a Test Execution Session (Data and Refs)

0. Exit

Enter your choice :

Figure 10–39 :*Selection of the ’Delete Execution Session’ Menu*

This menu contains all the recovery scripts operations related to the deletion of test execution sessions. It displays 3 possible actions.

- **Delete Default Test Session Data** (Session Refs remain) is selected by entering ’1’. This operation deletes all the data of the Default Test Session and leaves the session empty. The user is first asked to confirm the deletion request (see Figure 10–40). If he enters ’y’ or ’Y’, the Recovery Scripts delete all the data of the Default Test Session, keeping the session references.

```
 – 1.2 Delete Execution Session Menu –
________________________________________

1. Delete Default Test Session Data (Session Refs remain)
2. Delete the On–Line Data of an Archived Test Execution Session (Refs remain)
3. Delete Completely a Test Execution Session (Data and Refs)

0. Exit

Enter your choice : 1

1.2.1 Delete Default Test Session Data

        This operation deletes all the data of the DEFAULT_TEST_SESSION.
        The DEFAULT_TEST_SESSION remains open and empty.

        CAUTION: This action will delete all the data of the DEFAULT TEST SESSION.
        DEFAULT TEST SESSION will remain but empty.
        Do you want to continue (y/n) :
```

Figure 10–40 :*Selection of the ’Delete Default Test Session Data’ Operation.*

- **Delete the On–Line Data of an Archived Test Execution Session** (Refs remain) is selected by entering ’2’. An example is given at the Figure 10–41 The user wants to delete the on–line data of the archived session SESSION_QWERTY. He is first asked for the name of the archived session concerned. Secondly, he has to confirm the deletion request (see Figure 10–41). If he enters ’y’ or ’Y’, the Recovery Scripts delete all the on–line data of the given session and resets the session references accordingly. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern ’No such file of directory’ or ’no match’, they can be ignored.

```
– 1.2 Delete Execution Session Menu –
________________________________________

1. Delete Default Test Session Data (Session Refs remain)
2. Delete the On–Line Data of an Archived Test Execution Session (Refs remain)
3. Delete Completely a Test Execution Session (Data and Refs)

0. Exit

Enter your choice : 2


1.2.2 Delete the On–Line Data of an Archived Test Execution Session (Refs remain)

        This operation deletes the on–line data from the TRDB disk for
        a session which has been archived.  It updates the Oracle references
        accordingly (session state ARCH, no owner, files ARC).

        Enter session name (ONLY sessions that were archived): SESSION_QWERTY

        CAUTION: This action will delete all the on–line data of  SESSION_QWERTY.
        Be sure that the session data is archived on Final Archive (no automatic check).
        Do you want to continue (y/n) : █
```

Figure 10–41 :*Selection of 'Delete the On–Line Data of an Archived Session' Operation.*

- **Delete Completely a Test Execution Session (Data and Refs)** is selected by entering '3'. An example is given Figure 10–42. The user wants to delete completely the session SESSION_TEST. He is first asked for the name of the session concerned. Secondly, he has to confirm the deletion request (see Figure 10–42). If he enters 'y' or 'Y', the Recovery Scripts delete all the data and references of the given session. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern 'No such file of directory' or 'no match', they can be ignored (see Figure 10–42).

– 1.2 Delete Execution Session Menu –

1. Delete Default Test Session Data (Session Refs remain)
2. Delete the On–Line Data of an Archived Test Execution Session (Refs remain)
3. Delete Completely a Test Execution Session (Data and Refs)

0. Exit

Enter your choice : 3

1.2.3 Delete Completely a Test Execution Session (Data and Refs)

Enter session name : SESSION_TEST
CAUTION: This action will delete all the data of SESSION_TEST.
Do you want to continue (y/n) : y
– deleting session Oracle data –
– deleting session directories –
rm: $DBS_HOME/data/EXECUTION/SESSION_TEST/RAW_DATA: No such file or directory
– listing participating applications –
– removing list of participating applications –

Figure 10–42 :*Selection of the 'Delete Completely a Test Execution Session' Operation.*

10.3.1.1.3 Session is Used Menu

As shown in the Figure 10–43, if the selection in the 'Execution Session Menu' is '3', the 'Session is Used Menu' is displayed.

The TRDB could remain in an inconsistent state if an operation related to the Final Archive (ARCHIVING, RETRIEVING, IMPORT and EXPORT) or the deletion of a session fails. This menu contains all the recovery actions to save the TRDB from such a failure. It displays 5 possible items.

- **Import failure** is selected by entering '1'.This operation deletes the data and references of a session whose import from Final Archive failed. After this operation has been executed, an application can request another Import of the same session.

  The first step is a deletion of the temporary data created for the IMPORT and EXPORT operations. In case no such data remain, an error message 'No match' could appear.

  The second step is the listing of all the sessions referenced as 'to be imported'. The user is asked to enter the name of a session out of this list. Finally, he is asked to confirm the deletion request (see Figure 10–43). If he enters 'y' or 'Y', the Recovery Scripts delete all the data of the given session.

```
– 1.3 Session is Used Menu –
________________________________________

1. Import Failure
2. Export Failure
3. Archiving Failure
4. Retrieving Failure
5. Deletion Failure

0. Exit

Enter your choice : 1

1.3.1 Import Failure

        This operation deletes the temporary data of a session whose import has failed.

        a. Deletion of Import temporary directories
No match.
        b. Deletion of the session whose Import failed.
        The list of sessions recorded as "To Be Imported" is:
                Session EXEC_BEN_002 , Session state TBIM .

        Enter session name (a name from the list): EXEC_BEN_002

        CAUTION: This action will delete all the data of  EXEC_BEN_002.
        Do you want to continue (y/n) : █
```

Figure 10–43 :*Import from FA Failure, Execution Session.*

- **Export failure** is selected by entering ’2’.

  This operation deletes the temporary data created for the Export operations. It updates also the session state to its initial value for the session whose export on Final Archive failed. After this recovery action, an application can request another export of the same session

  The first step is the deletion of the temporary data created for the Export operations. In case no such data remain, an error message ’No match’ could appear (see Figure 10–44).

  The second step is the update of the session state for the session whose export on Final Archive failed. Its state has to be reset at its initial value (Note that this initial state is included in some of the DBS error messages related to Export failure). The Recovery Scripts display the listing of all the sessions referenced as ’to be exported’. The user is asked to enter the name of a session out of this list. If the name is valid, the session data diagnostic is launched (see Figure 10–44), and the Recovery Scripts propose an initial session state. Finally, the user has to enter the initial state of the session and is asked to confirm the update. If he enters ’y’ or ’Y’, the Recovery Scripts updates the state of the given session with the given value.

```
        1.3.2 Export Failure

        This operation deletes the temporary data created for all the exports
        of sessions. It allows to reset the session state of a session whose
        export has failed.

        a. Deletion of Export temporary directories
No match.
        b. Resetting state of a session whose Export failed.
        The list of sessions recorded as "To Be EXported" is:

        Session EXEC_BEN_002 , Session state TBEX .
        Session EXEC_BEN_003 , Session state TBEX .
Enter session name (a name from the list): EXEC_BEN_003
Running the Session Diagnostic to find initial STATE of the session:
______________________________________________________________________
                ...
                ...
1.8 Checking of Session data
______________________________________________________________________
                ...
                ...
Checking of Session data finished
______________________________________________________________________

        The session contains some files referenced as archived and only on FA.
        The session state should be ARCHIVED (ARCH).

        Enter the session STATE wished (CLOS, ONLI or ARCH):
        CLOS if session contains ONLY files referenced as Central Accessible and not on FA.
        ONLI if session contains ONLY files referenced as retrieved from FA.
        ARCH if session contains some files referenced as archived and only on FA.
        The session state proposed is: ARCH
ARCH
        CAUTION: This action will reset the STATE of  EXEC_BEN_003 to ARCH.
        Do you want to continue (y/n) :
```

Figure 10–44 :*Export on FA Failure, Execution Session.*

- **Archiving failure** is selected by entering ’3’.

  This operation analyses the status of a session whose archiving to Final Archive failed. It proposes a recovery scenario. After this recovery action, an application can request another archiving on the same session.

  An example is given at the Figure 10–45. The user wants to recover from a failure during the archiving of the session EXEC_BEN_003. The Recovery Scripts list the sessions recorded as ’To Be ARchived’. EXEC_BEN_003 is the only one. The user is prompted to

enter a session name. The session data diagnostic is launched on the given session (EXEC_BEN_003) and the Recovery Scripts propose a corrective action based on the diagnostic. The user has to confirm before the corrective scenario is executed. In the example, the session EXEC_BEN_003 contains only files referenced on a Final Archive medium. Its state should be ARCHIVED.

1.3.3 Archiving Failure

This operation analyses the data of a session whose archiving failed,
and proposes a recovery action.

The list of sessions recorded as "To Be ARchived" is:

Session EXEC_BEN_003 , Session state TBAR .
Enter the name of session to recover: EXEC_BEN_003
Running the Session Diagnostic to find a recovery scenario:

...
1.8 Checking of Session data

...
Checking of Session data finished

The session contains ONLY files referenced as archived and only on FA.
***Recovery scenario** : The session state is set to ARCHIVED (ARCH), the session data remaining on TRDB disk is deleted (it is also on FA). The data on FA is referenced.*

CAUTION: This scenario will reset the state of EXEC_BEN_003 to ARCH and delete the session data remaining on TRDB disk.
Do you want to continue (y/n) :

Figure 10–45 :*Archiving on FA Failure, Execution Session.*

- **Retrieving failure** is selected by entering '4'.

  This operation deletes the on–line data from the TRDB disk for a session whose retrieving from the Final Archive failed. The Oracle references are updated accordingly (data not on–line, on FA only). After this recovery action, an application can request another retrieving on the same session.

  An example is given Figure 10–46 The user wants to recover from a failure during the retrieving of data from the archived session EXEC_TEST. He is prompted to enter a session name (EXEC_TEST is entered). He has to confirm the on–line data deletion for the given session. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern 'No such file of directory' or 'No match', they can be ignored.

1.3.4 Retrieving Failure

This operation deletes the on–line data from the TRDB disk for a session whose retrieving failed. It updates the Oracle references accordingly.

Enter session name: EXEC_TEST
CAUTION: This action will delete the on–line data from the TRDB disk for EXEC_TEST.
Do you want to continue (y/n) : y
– deleting the on–line data –
No match.
No match.
– resetting the Oracle references –

It is adviced to run the session diagnostic

Figure 10–46 :*Retrieving from FA Failure, Execution Session.*

- **Deletion failure** is selected by entering ’5’.

  This operation helps to recover from a deletion failure:
  – the deletion of the on–line data of an archived session (partial deletion) or
  – the complete deletion of all data and references of a closed session.
  It completes the required deletion.

  An example is given Figure 10–47 The user wants to recover from a failure during the partial deletion of the archived session EXEC_BEN_002. He is prompted to enter a session name (EXEC_BEN_002 is entered). He has to confirm the on–line data deletion. for the given archived session. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern ’No such file of directory’ or ’No match’, they can be ignored.

```
            1.3.5 Deletion Failure

            This operation deletes the session data from the TRDB disk for
             a session whose deletion failed.  The deletion can be complete
            (data and references) or partial (only data, for archived sessions)
            The list of sessions recorded as "To Be Deleted" is :

                   Session EXEC_BEN_002 , Session state TBPD .
            Enter session name: EXEC_BEN_002
            Do you want :
            1. a partial deletion (deletion of on–line data, for archived session),
            2. a complete deletion (on–line data and references)
1
            CAUTION: This action will delete all the on–line data of  EXEC_BEN_002.
            Do you want to continue (y/n) : y
   – deleting the on–line data –
No match
            ______________________________________________
             It is adviced to run the session diagnostic
            ______________________________________________
```

Figure 10–47 :*Deletion Failure, Execution Session.*

10.3.1.1.4 File Storage Failure Menu

If the selection in the 'Execution Session Menu' is '4', the 'File Storage Failure' menu is displayed, as shown in Figure 10–48 This menu contains all the recovery operations related to failure during storage of execution session files.

– 1. Execution Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Close Session
6. List Sessions
7. File Transfer from Local Dir Failures
8. Close aborted test sessions to NCL

0. Exit

Enter your choice : 4

1.4 File Storage Failure

– 1.4. File Storage Failure Menu –

1. List Local Files Remaining Unstored
2. Remove Local Files Remaining Unstored

0. Exit

Enter your choice :

Figure 10–48 :*File Storage Failure, Execution Session.*

- **List Local Files Remaining Unstored** is selected by entering ’1’.This operation executes the session data diagnostic for a given session and display a summary of the local unstored files (Local Accessible and Local Not Accessible files).

  An example is given at the figure Figure 10–49 The user wants to list the local files for the session EXEC_BEN_002. He is asked to enter a session name (see Figure 10–49).

– 1.4. File Storage Failure Menu –

1. List Local Files Remaining Unstored
2. Remove Local Files Remaining Unstored

0. Exit

Enter your choice : 1

1.4.1 List Files Remaining Unstored

This operation calls the session data diagnostic to list files unstored.

Enter session name : EXEC_BEN_002

Unstored_files Execution Session Diagnostic : <EXEC_BEN_002>

...

1.2 Listing of Local Accessible files referenced in the TRDB

Reference of File found : /ariane6/projects/VICOS/rt_user3/qwerty22
Listing of Local Accessible files finished :1 file(s)

...

1.8 Checking of Session data

Session is recorded as CLOSED
Only Central Accessible files should exists.
1 file(s) is(are) referenced as local accessible
** These files and their references should be removed.*
Checking of Session data finished

1 Local Accessible files have to be deleted
0 Local Not Accessible files have to be deleted

Figure 10–49 :*List Local Files Unstored, Execution Session.*

- **Remove Local Files Remaining Unstored** is selected by entering ’2’. This operation deletes the Oracle references to the unstored files of a given session (Local Accessible and Local Not Accessible files).

  An example is given Figure 10–50 The user wants to recover from a failure during the storage of a file for session EXEC_BEN_002. The user is asked to enter a session name (see Figure 10–50) and can specify a file type. He has to confirm the deletion. **The flat files corresponding to the references deleted have to be removed manually from their location**.

```
– 1.4. File Storage Failure Menu –
________________________________

1. List Local Files Remaining Unstored
2. Remove Local Files Remaining Unstored

0. Exit

Enter your choice : 2

        1.4.2 Remove Local Files Remaining Unstored

        This operation removes the Oracle references to Local Files Remaining Unstored.
        The flat files corresponding have to be deleted manually.

        Enter session name : EXEC_002
        Enter file type (RD / RESULT / EVENT / EVL / CONFIG / ALL) : RD
        CAUTION: This action will delete the Oracle refs to  RD Local Files of  EXEC_002.
        Do you want to continue (y/n) :  █
```

Figure 10–50 :*Remove Local Files Unstored, Execution Session.*

10.3.1.1.5 Close Session

If the selection in the ’Execution Session Menu’ is ’5’, the ’Close Session’ operation is executed.

This operation allows the user to force the closure of a session whose normal closure failed. An example is given Figure 10–51 The user wants to recover from a failure during the closure of EXEC_BEN_007. It starts by the listing of all the open sessions. The user is asked to enter a session name out of the list and to confirm the action. If he enters ’y’ or ’Y’, the Recovery Scripts will:

- set the session state to CLOSED (CLOS),
- remove the references to participating applications to warn applications that the session is closed and
- set the temporary EVL files accessible.

```
 – 1. Execution Session Menu –
_____________________________
1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Close Session
6. List Sessions
7. File Transfer from Local Dir Failures
8. Close aborted test sessions to NCL

0. Exit
Enter your choice : 5
        1.5 Close Session

        This operation forces the closure of a session whose normal closure failed.
        It sets the session state to CLOSED (CLOS), removes the references
        for the participating applications and sets temporary EVL files accessible.
        The list of sessions recorded as ”OPEN” is:

                Session DEFAULT_TEST_SESSION , Session state OPEN .
                Session EXEC_BEN_007 , Session state OPEN .
                Session EXEC_BEN_008 , Session state OPEN .
        Enter session name (a name from the list): EXEC_BEN_007
        CAUTION: This action will force the closure  of  EXEC_BEN_007.
        Do you want to continue (y/n) : y
        – set session state to CLOS –
        – listing participating applications –
          TES_01 participating in session EXEC_BEN_008
                remains ...
          TES_07 participating in session EXEC_BEN_007
                removed ...
          TES_25 participating in session EXEC_BEN_007
                removed ...
        – removing list of participating applications –
        – set temporary EVL files accessible –
```

Figure 10–51 :*Close Execution Session.*

10.3.1.1.6 List Execution Sessions

If the selection in the ’Execution Session Menu’ is ’6’, the ’List Sessions’ operation is executed.

This operation allows the user to list all the execution sessions recorded in the TRDB. The data displayed contain the session name, the session state, the test mode, the closure status, the session owner and the session creator (see Figure 10–52).

– 1. Execution Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Close Session
6. List Sessions
7. File Transfer from Local Dir Failures
8. Close aborted test sessions to NCL

0. Exit

Enter your choice : 6

1.6 List Sessions

This operation lists all the execution sessions
recorded in the TRDB. The data displayed is :
the session name, the session state (OPEN, CLOSed, ARCHived, ONLIne),
the test mode (1 character), the closure status (normally closed,
aborted, open), the session owner and the session creator.

NAME	*STAT*	*T*	*CLO*	*OWNER*	*CREATOR*
IMPORTED	*CLOS*	*A*	*NCL*	*rt_user3*	*rt_user3*
AFTER_IMPORT	*CLOS*	*A*	*NCL*	*rt_user3*	*rt_user3*
TEST_ARC1	*ARCH*	*A*	*NCL*		*rt_user3*
TEST_ARC2	*ARCH*	*A*	*NCL*		*rt_user3*
TEST_ARC3	*ARCH*	*A*	*NCL*		*rt_user3*
TEST_ARC4	*ARCH*	*A*	*NCL*		*rt_user3*
TEST_EVL	*CLOS*	*A*	*ABT*	*rt_user2*	*rt_user2*
DEFAULT_TEST_SESSION	*OPEN*	*N*	*OPN*		
EXEC_002	*ARCH*	*A*	*NCL*	*rt_user7*	*rt_user3*
EXEC_003	*ARCH*	*A*	*NCL*		*rt_user3*
EXEC_004	*CLOS*	*A*	*NCL*	*rt_user3*	*rt_user3*
EXEC_005	*ARCH*	*A*	*NCL*	*rt_user3*	*rt_user3*

12 rows selected.

Figure 10–52 : *Execution Session List.*

10.3.1.1.7 File Transfer from Local Dir Failures

If the selection in the 'Execution Session Menu' is '7', a menu is provided which allows to handle failures in transferring archive or event files from local disks to the central TRDB's disks.

Having selected the test session (menu entry 1), the current files of a session may be listed or the local files may be transfered to the TRDB's central disk. Each node (testnode, workstation) is to be handled separately. The location of the local files have to be specified by the user.

– 1. Execution Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Close Session
6. List Sessions
7. File Transfer from Local Dir Failures
8. Close aborted test sessions to NCL

0. Exit

Enter your choice : 7

1.7 File Transfer from Local Dir Failures

Recover from incomplete/not executed file transfers from local
DBS/archive directories (on test nodes or workstations)
Recovers EVT, EVL or archive files and adds them to a test session)

Recover Session Files Menu –

1. Recover files for multiple sessions (auto detection)
2. Specify Test Session
3. Recover all archive files (each node separately)
4. Recover event/evl files (each node separately)
5. List all files in Test Session
6. List Contents of Archive Files in Test Session (TOC: Pathnames)
7. List Items of Archive Files in Test Session (all archived entries)

0. Exit

Enter your choice :

Figure 10–53 : *File Transfer Failures.*

Recover files for multiple sessions (auto detection) (Selection: 1)

This selection allows for recovery of files with ”auto detection” of the related test session, i.e. according to the creation date of the found files they will automatically be assigned to the correct testsession and be stored in the scope of that session. The following submenu is displayed:

Enter Testnode directory............................ 1
Analyze only... 2 [YES]
Define Logfile.. 3
Delete files on testnode after restore......... 4 [YES]
Assign unrelated files to specific session.. 5
Continue in case of error.......................... 6 [NO]
Analyze files... 7
List sessions... 8
Exit... 9

Enter number of command:

Figure 10–54 :*Recover files for multiple sessions with auto–detection*

Command description:

1. Enter Testnode directory
   The line ”Enter eventfile directory:” is displayed. The name of the testnode directory is expected here, e.g. /testnode/cgs–test. This name will be displayed below menu item [1]:
   Enter Testnode directory.......................... 1
   [/testnode/cgs–test]
   Analyze only.. 2 [YES]

2. Analyze only
   This command will have an influence on item [7], Analyze/Restore Files. If selected the value ”YES” displayed at the end of the line will switch to ”NO” and the line for menu item [7] will change from ”Analyze files” to ”Restore files” (see below).

3. Define Logfile
   If the question ”Store messages in file (y/n)? [n]” is answered with ”y” a logfile can be entered to contain all messages displayed in the scope of this recovery. If the question to enter the name of the logfile is simply answered with a <return> a default logfile named ”STORE_FILES.LOG” will be created in the current directory. The selected name will be displayed below menu item [3]:
   Define Logfile.. 3
   [STORE_FILES.LOG]
   Delete files on testnode after restore............ 4 [YES]

4. Delete files on testnode after restore
   On selection of this item the value ”YES” at the end of the line switches to ”NO”, i.e. the files will remain on the selected testnode directory after recovery.

5. Assign unrelated files to specific session
   If a file cannot be assigned to a testsession it may be related to a specific session. On selection of this item the name of that session can be entered:
   Add unrelated files to session (DEFAULT_TEST_SESSION):
   (default: DEFAULT_TEST_SESSION, selected on <return>)

The selected name will be displayed below menu item [5]:

Assign unrelated files to specific session........ 5
[DEFAULT_TEST_SESSION]
Continue in case of error..................................... 6 [NO]

Note that files which cannot be assigned to a session will remain on the local directory in case no session has been selected alternatively.

6. Continue in case of error
In case of any error during the recovery process the activity is stopped immediately. This selection will force the system to continue in case of errors.

7. Analyze files/Restore files
In case the selection of item [2], Analyze only is set to ”YES” the files found on the specified testnode directory will be analyzed, i.e. the name of the found files is displayed together with the related testsession. If no session can be assigned this will be displayed accordingly:

```
Eventfiles:
  /testnode/cgs–test/TSCV_01.28–03–00_10:26:48:597.EVT.002...assigned to session:TEST_01
   /testnode/cgs–test/HCI_01.28–03–00_10:30:34:680.EVT.007...assigned to session:TEST_02
EVL files:
 /testnode/cgs–test/TSCV_01.28–03–00_10:26:48:619.EVL.003...assigned to session:TEST_01
  /testnode/cgs–test/HCI_01.28–03–00_10:29:49:531.EVL.003...not related to any session
```

In case the selection of item [2], Analyze only is set to ”NO” the files found on the specified testnode directory will be stored in the TRDB in the scope of the assigned session.

8. List sessions
This will display a list of all sessions.

9. Exit
This selection will end the recovery session and will bring the user back to the previous menu.

Specify Test Session (Selection: 2)

The name of the test session which is to be recovered is to be entered. It is applicable for all subsequent selections.

An existing execution session must be specified. The session may be in any state.

Recover all archive files (Selection: 3)

For each node, the test node's local directory must be specified (without subdirectory "archive")

After succesful transfer, the files should be deleted in the local directory.

```
Enter your choice : 2
Enter Test Node directory ([/testnode/host_name]): /testnode/cgs–test
–rw–r––r––   1 cgsadmin cgs       11264 May  7 18:18 TES_01001_199903111753.arc
–rw–r––r––   1 cgsadmin cgs      113664 May  7 18:18 TES_01002_199903111754.arc

Shall all of these files be added to the test session PAUL_RECOVERED  (y/[n]) ? y

...

Check that the files have been registered and copied to the test session
Shall they all be deleted now on the local disk ? (y/[n]) y
```

Figure 10–55 : *Recover Archive Files for one test node*

Recover event/evl files (Selection: 4)

For each test node, the node's local directory must be specified
For each workstation, the DBS work directory must be specified.

```
Enter your choice: 3
Default Location: /gsaf_home/dbs/data/WORK/DEFAULT_TEST_SESSION
On test nodes: /testnode/<host_name>

Enter event/evl file directory to be added:

/gsaf_home/dbs/data/WORK/DEFAULT_TEST_SESSION

No match

Note: EVL files will be deleted directly after insertion
Shall all of these files be added to the test session PAUL_RECOVERED  (y/[n]) ?n

cleanup directory /gsaf_home/dbs/data/WORK/DEFAULT_TEST_SESSION and start again
```

Figure 10–56 : *Recover Event/Evl Files for one test node*

List all files in test session (Selection: 5)

The files already transfered and stored currently on the DB Server’s central disk are listed.

List contents of archive files in test session / Pathames (Selection: 6)

The archive files already stored within the test session are scanned for items stored. A list is generated for each archive item type (SMT Updates, GDUs, ADUs) and for all types together.
The list shows each SID and pathname stored (at least once) in the files.
The lists are generated as ASCII files and stored under <test_session>.<type> resp. <test_session>.lst
They are presented to the user via the OpenLook Texteditor.

List contents of archive files in test session / Entries (Selection: 7)

The archive files already stored within the test session are scanned for items stored. A list of all items together with their timetag is generated for each archive item type (SMT Updates, GDUs, ADUs) and for all types together.
The lists are generated as ASCII files and stored under <test_session>.<type> resp. <test_session>.lst
They are presented to the user via the OpenLook Texteditor.

10.3.1.1.8 Close Aborted Test Session to Normally Closed

If the selection in the ’Execution Session Menu’ is ’8’, a function is provided which allows to close an aborted test session to the state NORMALLY CLOSED, thus allowing to export the session and to evaluate the session as usual.

Note: This function should be called **carefully:** The contents of the session might be corrupted! When setting the session state to NORMALLY CLOSED, no indication is availalbe anymore that something has gone wrong during execution of the session.

```
          – 1. Execution Session Menu –
          ______________________________

          1. Session Data Diagnostic
          2. Delete Session
          3. Session is Used
          4. File Storage Failure
          5. Close Session
          6. List Sessions
          7. File Transfer from Local Dir Failures
          8. Close Aborted Test Sessions to NCL

           0. Exit

           Enter your choice : 8

 1.8 Close Aborted Test Sessions to Normally Closed

 Allow for evaluation/final archiving of aborted test sessions
by setting it to normally closed
Should only be called for test sessions which cannot be handled otherwise

   >>> Warning: The test session will appear as normally closed session         <<<
   >>>           Even if there are inconsistencies of any kind within the session  <<<
  <list of sessions>
  Enter session name (a name from the list): <SESSION>

 CAUTION: This action will force the normal closure  of <SESSION>
          The session status will not indicate anymore any problem during closure
          The session cannot be set to ABORTED again

   Do you want to continue (y/n) : y
```

Figure 10–57 : *Set aborted session to normally closed.*

10.3.1.2 Evaluation Session Menu

Select the Evaluation Session Menu by entering ’2’ in the Main Menu. The Evaluation Session Menu allows to call all the recovery scripts operations related to test evaluation sessions. It is shown in figure Figure 10–58 The user has to chose between 5 items. The choice 0 returns to the Main Menu.

– DBS RECOVERY SCRIPTS MAIN MENU –

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_ERR_xxx)

0. Exit

Enter your choice : 2

– 2. Evaluation Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Remove Evaluation Users
6. List Sessions

0. Exit

Enter your choice : 1

2.1 Session Data Diagnostic

Enter session name : █

Figure 10–58 :*Selection of the Evaluation Session Menu from the Main Menu*

10.3.1.2.1 Evaluation Session Diagnostic

As shown in the Figure 10–58, if the selection in the Evaluation Session Menu is ’1’, the user is asked for the session whose data diagnostic is required.

This functionality allows the scanning of the session data references and the display of a report. The information provided summarizes the session status, and helps the user to decide which recovery action is required. The Figure 10–59 shows such a report for a session named EVAL_01 which is retrieved from the Final Archive.

A report contains 6 sections. The 5 first sections list the references of the session data files. The division is based on their FILE_STATUS :

- Central ACcessible (CAC) files: these are files normally stored into the TRDB, accessible for evaluation ('on–line').
- Local ACcessible (LAC) files: these are files that local applications expect to be stored into the TRDB by Central DBS; they are not yet accessible for evaluation. This state is normally temporary during the storage into the TRDB by Central DBS (OPEN sessions). A file whose storage into the TRDB failed may remain in the state Local Accessible
- Central Not Accessible (CNA) files: these are files managed by Central DBS but not accessible for evaluation. It can be an error case.
- Local Not Accessible (LNA) files: these are files not accessible for evaluation and to Central DBS. It is an error case.
- ARChived (ARC) files: these are files archived on an Final Archive medium and not retrieved on the TRDB disk (not on–line).

The last section displays the session state. It checks the consistency of the session data state compared with the session state and summarize the situation.

The last report generated by the Recovery Scripts is maintained into the file ***$DBS_HOME/util/sun5/recovery/work/Recovery.diagnostic***.

Recovery Evaluation Session Diagnostic : <EVAL_01>

The diagnostic analyses the TRDB references for session EVAL_01
i.e. the filename location, the session state/session data state.
Files are classified as :
– Central Accessible (normally stored into the TRDB, accessible for evaluation),
– Local Accessible (files that Local nodes expect to store into TRDB),
– Central Not Accessible (files that are not accessible for evaluation),
– Local Not Accessible (files that are not accessible for evaluation and to Central),
– Archived (files that are archived on an FA SAS Medium).
The latest diagnostic is kept into a file: $DBS_HOME/util/sun5/recovery/work/Recovery.diagnostic

1.1 Listing of Central Accessible files referenced in the TRDB

Reference of File found: $DBS_HOME/data/EVALUATION/EVAL_01/RESULT/GRAPH/q2
Reference of File found: $DBS_HOME/data/EVALUATION/EVAL_01/RESULT/REPORT/qq
Listing of Central Accessible files finished :2 file(s)

1.2 Listing of Local Accessible files referenced in the TRDB

No Local Accessible file referenced in TRDB
Listing of Local Accessible files finished :0 file(s)

1.3 Listing of Central NOT Accessible files referenced in the TRDB

No Central NOT Accessible file referenced in TRDB
Listing of NOT Accessible files finished :0 file(s)

1.4 Listing of Local NOT Accessible files referenced in the TRDB

No Local NOT Accessible file referenced in TRDB
Listing of NOT Local Accessible files finished :0 file(s)

1.5 Listing of Archived files referenced in the TRDB

No Archived file referenced in TRDB
Listing of Archived files finished :0 file(s)

Reference of Retrieved File found: $DBS_HOME/data/EVALUATION/ EVAL_01/RESULT/GRAPH/q2
(should match a Central Accessible Reference).
Disk name is : MOD*ARCH*15.02.96–17:35
Reference of Retrieved File found: $DBS_HOME/data/EVALUATION/EVAL_01/RESULT/REPORT/qq
(should match a Central Accessible Reference).
Disk name is : MOD*ARCH*15.02.96–17:35
Listing of Retrieved files finished :2 file(s)

1.6 Checking of Session data

Session is recorded as ON–LINE
All data should be Central Accessible, retrieved from FA.
2 file(s) is(are) referenced as retrieved from FA and Central Accessible
Checking of Session data finished

Figure 10–59 :*Report of the Evaluation Session Data Diagnostic*

10.3.1.2.2 Delete Evaluation Session Menu

If the selection in the Evaluation Session Menu is ’2’, the Delete Evaluation Session Menu is displayed.

This menu contains all the recovery scripts operations related to the deletion of evaluation sessions. It displays 2 possible actions.

- **Delete the On–Line Data of an Archived Evaluation Session** (Refs remain) is selected by entering ’1’. An example is given Figure 10–60 The user wants to recover from a failure during the partial deletion of EVAL_001. He is first asked for the name of the archived session concerned. Secondly, he has to confirm the deletion request (see Figure 10–60). If he enters ’y’ or ’Y’, the Recovery Scripts delete all the on–line data of the given session and resets the session references accordingly. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern ’No such file of directory’ or ’no match’, they can be ignored.

– 2.2 Delete Evaluation Session Menu –

1. Delete the On–Line Data of an Archived Evaluation Session (Refs remain)
2. Delete Completely an Evaluation Session (Data and Refs)

0. Exit

Enter your choice : 1

2.2.1 Delete the On–Line Data of an Archived Evaluation Session (Refs remain)

This operation deletes the on–line data from the TRDB disk for
a session which has been archived. It updates the Oracle references
accordingly (session state ARCH, no owner, files ARC).

Enter session name (ONLY sessions that were archived): EVAL_001
CAUTION: This action will delete all the on–line data of EVAL_001.
Be sure that the session data is archived on Final Archive (no automatic check).
Do you want to continue (y/n) : y
– deleting the on–line data –
– resetting the Oracle references –

It is adviced to run the session diagnostic

Figure 10–60 :*Selection of ’Delete the On–Line Data of an Archived Session’, Evaluation Session.*

- **Delete Completely an Evaluation Session (Data and Refs)** is selected by entering ’2’. An example is given Figure 10–61 The user wants to recover from a failure during the complete deletion of EVAL_004. He is first asked for the name of the session concerned. Secondly, he has to confirm the deletion request (see Figure 10–61). If he enters ’y’ or ’Y’, the Recovery Scripts delete all the data and references of the given session. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern ’No such file of directory’ or ’no match’, they can be ignored.

– 2.2 Delete Evaluation Session Menu –

1. Delete the On–Line Data of an Archived Evaluation Session (Refs remain)
2. Delete Completely an Evaluation Session (Data and Refs)

0. Exit

Enter your choice : 2

2.2.2 Delete Completely an Evaluation Session (Data and Refs)

Enter session name : EVAL_004
CAUTION: This action will delete all the data of EVAL_004.
Do you want to continue (y/n) : y
– deleting session Oracle data –
– deleting session directories –

Figure 10–61 :*Selection of 'Delete Completely an Evaluation Session' Operation.*

10.3.1.2.3 Session is Used Menu

If the selection in the 'Evaluation Session Menu' is '3', the 'Session is Used Menu' is displayed.

The TRDB could remain in an inconsistent state if an operation related to the Final Archive (ARCHIVING, RETRIEVING, IMPORT and EXPORT) or the deletion of a session fails. This menu contains all the recovery actions to recover the TRDB from such a failure. It displays 5 possible items.

- **Import failure** is selected by entering '1'.This operation deletes the data and references of a session whose import from Final Archive failed. After this operation has been executed, an application can request another Import of the same session.

  The first step is a deletion of the temporary data created for the IMPORT and EXPORT operations. In case no such data remain, an error message 'No match' could appear. It can be ignored.

  The second step is the listing of all the evaluation sessions referenced as 'to be imported'. The user is asked to enter the name of a session out of this list. Finally, he is asked to confirm the deletion request (see Figure 10–62). If he enters 'y' or 'Y', the Recovery Scripts delete all the data of the given session.

– 2.3 Session is Used Menu –

1. Import Failure
2. Export Failure
3. Archiving Failure
4. Retrieving Failure
5. Deletion Failure

0. Exit

Enter your choice : 1

2.3.1 Import Failure

This operation deletes the temporary data of a session whose import has failed.

a. Deletion of Import temporary directories
No match.
b. Deletion of the session whose Import failed.
The list of sessions recorded as ”To Be Imported” is:

Session EVAL_004 , Session state TBIM .
Enter session name (a name from the list): EVAL_004
CAUTION: This action will delete all the data of EVAL_004.
Do you want to continue (y/n) : y
– deleting session Oracle data –
– deleting session directories –

It is adviced to run the session diagnostic

Figure 10–62 :*Import from FA Failure, Evaluation Session.*

- **Export failure** is selected by entering ’2’.

  This operation deletes the temporary data created for the Export operations. It updates also the session state to its initial value for the session whose export on Final Archive failed. After this recovery action, an application can request another export of the same session

  The first step is the deletion of the temporary data created for the Export operations. In case no such data remain, an error message ’No match’ could appear (see Figure 10–63).

  The second step is the update of the session state for the session whose export on Final Archive failed. Its state has to be reset at its initial value (Note that this initial state is included in some of the DBS error messages related to Export failure). The Recovery Scripts display the listing of all the sessions referenced as ’to be exported’. The user is asked to enter the name of a session out of this list. If the name is valid, the session data diagnostic is launched

(see Figure 10–63), and the Recovery Scripts propose an initial session state. Finally, the user has to enter the initial state of the session and is asked to confirm the update. If he enters ’y’ or ’Y’, the Recovery Scripts updates the state of the given session with the given value.

2.3.2 Export Failure

This operation deletes the temporary data created for all the exports of sessions. It allows to reset the session state of a session whose export has failed.

a. Deletion of Export temporary directories
No match.
b. Resetting state of a session whose Export failed.
The list of sessions recorded as ”To Be EXported” is:

Session EVAL_002 , Session state TBEX .
Enter session name (a name from the list): EVAL_002
Running the Session Diagnostic to find initial STATE of the session:

Session_is_Used Evaluation Session Diagnostic : <EVAL_002>
...
1.6 Checking of Session data

...
Checking of Session data finished

The session contains ONLY files referenced as Central Accessible and not on FA.
The session state should be CREATED (CREA).

Enter the session STATE wished (CREA, ONLI or ARCH):
CREA if session contains ONLY files referenced as Central Accessible and not on FA.
ONLI if session contains ONLY files referenced as retrieved from FA.
ARCH if session contains ONLY files referenced as archived and only on FA.
The session state proposed is: CREA
CREA
CAUTION: This action will reset the STATE of EVAL_BEN_002 to CREA.
Do you want to continue (y/n) : y
– resetting session state –

It is adviced to run the session diagnostic

Figure 10–63 :*Export on FA Failure, Evaluation Session.*

- **Archiving failure** is selected by entering ’3’.

This operation analyses the status of a session whose archiving to Final Archive failed. It proposes a recovery scenario. After this recovery action, an application can request another

archiving on the same session.

An example is given Figure 10–64 The user wants to recover from a failure during the archiving of EVAL_002 on FA. The Recovery Scripts list the sessions recorded as 'To Be ARchived'. EVAL_002 is the only one. The user is prompted to enter a session name. The session data diagnostic is launched on the given session (EVAL_002) and the Recovery Scripts propose a corrective action based on the diagnostic. The user has to confirm before the corrective scenario is executed. In the example, the session EVAL_002 contains files referenced only on the TRDB disk. Its state should be CREATED (CREA).

2.3.3 Archiving Failure

This operation analyses the data of a session whose archiving failed,
and proposes a recovery action.

The list of sessions recorded as "To Be ARchived" is:

Session EVAL_002 , Session state TBAR .
Enter the name of session to recover: EVAL_002
Running the Session Diagnostic to find a recovery scenario:

Session_is_Used Evaluation Session Diagnostic : <EVAL_002>

...

1.6 Checking of Session data

Session is recorded as To Be Archived

...

Checking of Session data finished

The session contains ONLY files referenced as Central Accessible and not on FA.
These files are all accessible.
Recovery scenario : The session state is set to CREATED (CREA).

CAUTION: This scenario will reset the state of EVAL_002 to CREATED.
Do you want to continue (y/n) : █

Figure 10–64 :*Archiving on FA Failure, Evaluation Session.*

- **Retrieving failure** is selected by entering '4'.

This operation deletes the on–line data from the TRDB disk for a session whose retrieving from the Final Archive failed. The Oracle references are updated accordingly (data not on–line, on FA only). For a retrieving with a NEW_NAME, the **Import Failure** operation (see 10.3.1.2.3.) must be called afterwards to complete the corrective actions. After this, an application can request another retrieving on the same session.

Two examples are given at the figure Figure 10–65 and Figure 10–66 In the first one, the user wants to recover from a failure during the retrieving of EVAL_001 with a null

NEW_NAME. The user is prompted to enter a session name (EVAL_001 is entered). He has to confirm the on–line data deletion for the given session. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern ’No such file of directory’ or ’No match’, they can be ignored.

In the second example (see Figure 10–66), the user wants to recover from a failure during the retrieving of EVAL_001 with NEW_NAME = EVAL_002. He has called the ’Session is Used’ menu, and has chosen the fourth option ’Retrieving Failure’. The Recovery Scripts ask for a session name (EVAL_001 is entered). The user has to confirm the on–line data deletion for the given session. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern ’No such file of directory’ or ’No match’, they can be ignored. After completion of the operation, the menu ’Session is Used’ is displayed. The user selects the first option ’Import Failure’ and deletes the temporary data of the session EVAL_002.

```
2.3.4 Retrieving Failure

This operation deletes the on–line data from the TRDB disk for
a session whose retrieving failed.  It updates the Oracle references
accordingly.
For a retrieving with a NEW_NAME, the Menu <Import Failure> must
be used afterwards to delete the NEW_NAME session.

Enter session name: EVAL_BEN_001
CAUTION: This action will delete the on–line data from TRDB disk for  EVAL_001
Do you want to continue (y/n) : y
          – deleting the on–line data –
          – resetting the Oracle references –
____________________________________________________
           It is adviced to run the session diagnostic
____________________________________________________
```

Figure 10–65 :*Retrieving from FA Failure, Evaluation Session, no NEW_NAME.*

2.3.4 Retrieving Failure

This operation deletes the on–line data from the TRDB disk for a session whose retrieving failed. It updates the Oracle references accordingly.
For a retrieving with a NEW_NAME, the Menu <Import Failure> must be used afterwards to delete the NEW_NAME session.

Enter session name: EVAL_BEN_001
CAUTION: This action will delete the on–line data from TRDB disk for EVAL_001
Do you want to continue (y/n) : y
– deleting the on–line data –
– resetting the Oracle references –

It is adviced to run the session diagnostic

– 2.3 Session is Used Menu –

---

1. Import Failure
2. Export Failure
3. Archiving Failure
4. Retrieving Failure
5. Deletion Failure

0. Exit

Enter your choice : 1

2.3.1 Import Failure

This operation deletes the temporary data of a session whose import has failed.
a. Deletion of Import temporary directories
No match.
b. Deletion of the session whose Import failed.
The list of sessions recorded as ”To Be Imported” is:

Session EVAL_002 , Session state TBIM .
Enter session name (a name from the list): EVAL_002
CAUTION: This action will delete all the data of EVAL_002.
Do you want to continue (y/n) : y
– deleting session Oracle data –
– deleting session directories –

It is adviced to run the session diagnostic

Figure 10–66 :*Retrieving from FA Failure, Evaluation Session, NEW_NAME not null.*

- **Deletion failure** is selected by entering ’5’.

  This operation helps to recover from a deletion failure:
  – the deletion of the on–line data of an archived session (partial deletion) or
  – the complete deletion of all data and references of a closed session.
  It completes the required deletion.

  An example is given Figure 10–67 The user wants to recover from a failure in the complete deletion of EVAL_002. He is prompted to enter a session name (EVAL_002 is entered). He has to confirm the session deletion. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern ’No such file of directory’ or ’No match’, they can be ignored.

```
        2.3.5 Deletion Failure

        This operation deletes the session data from the TRDB disk for
        a session whose deletion failed.
        The list of sessions recorded as ”To Be Deleted” is :

                Session EVAL_002 , Session state TBCD .
        Enter session name: EVAL_002
        Do you want :
        1. a partial deletion (deletion of on–line data, for archived session),
        2. a complete deletion (on–line data and references)
2
        CAUTION: This action will delete all the data of  EVAL_BEN_002.
        Do you want to continue (y/n) : y
    – deleting session Oracle data –
    – deleting session directories –
```

Figure 10–67 :*Deletion Failure, Evaluation Session.*

10.3.1.2.4 File Storage Failure Menu

If the selection in the ’Evaluation Session Menu’ is ’4’, the ’File Storage Failure’ menu is displayed, as shown in Figure 10–68. This menu contains all the recovery operations related to failure during storage of evaluation session files.

```
– 2. Evaluation Session Menu –
________________________________

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Remove Evaluation Users
6. List Sessions

0. Exit

Enter your choice : 4

2.4 File Storage Failure

        – 2.4. File Storage Failure Menu –
        ___________________________________

        1. List Local Files Remaining Unstored
        2. Remove Local Files Remaining Unstored

        0. Exit

        Enter your choice : █
```

Figure 10–68 :*File Storage Failure, Evaluation Session.*

It displays 2 possible items.

- **List Local Files Remaining Unstored** is selected by entering ’1’.This operation executes the session data diagnostic for a given session and display a summary of the local unstored files (Local Accessible and Local Not Accessible files).

  An example is given at the figure Figure 10–69 The user wants to list the local files for the session EVAL_002. He is asked to enter a session name (see Figure 10–69).

– 2.4. File Storage Failure Menu –

1. List Local Files Remaining Unstored
2. Remove Local Files Remaining Unstored

0. Exit

Enter your choice : 1

2.4.1 List Files Remaining Unstored

This operation calls the session data diagnostic to
list files unstored.

Enter session name : EVAL_002

Unstored_files Evaluation Session Diagnostic : <EVAL_002>

...

1.2 Listing of Local Accessible files referenced in the TRDB

Reference of File found : /ariane6/projects/VICOS/rt_user3/qwerty22
Listing of Local Accessible files finished :1 file(s)

...

1.8 Checking of Session data

Session is recorded as CREATED (on–line and not archived)
1 file(s) is(are) referenced as local accessible
** Note that at least one file has not been found at its referenced location.*
Checking of Session data finished

1 Local Accessible files have to be deleted
0 Local Not Accessible files have to be deleted

Figure 10–69 :*List Local Files Unstored, Evaluation Session.*

- **Remove Local Files Remaining Unstored** is selected by entering ’2’. This operation deletes the Oracle references to the unstored files of a given session (Local Accessible and Local Not Accessible files).

  An example is given Figure 10–70. The user wants to recover from a failure during the storage of a file for session EVAL_002. The user is asked to enter a session name (see Figure 10–70) and to confirm the deletion. **The flat files corresponding to the references deleted have to be removed manually from their location**.

– 2.4. File Storage Failure Menu –

1. List Local Files Remaining Unstored
2. Remove Local Files Remaining Unstored

0. Exit

Enter your choice : 2

2.4.2 Remove Local Files Remaining Unstored

This operation removes the Oracle references to Local Files Remaining Unstored.
The flat files corresponding have to be deleted manually.

Enter session name : EVAL_002
CAUTION: This action will delete the Oracle refs to Local Files of EVAL_002.
Do you want to continue (y/n) :

Figure 10–70 :*Remove Local Files Unstored, Evaluation Session.*

10.3.1.2.5 Remove Evaluation Users Menu

If the selection in the ’Evaluation Session Menu’ is ’5’, the ’Remove Evaluation Users’ menu is selected. This menu (see Figure 10–71) provides the operations related to evaluation user references, i.e.

- a reference for the evaluation connection,
- references of sessions allocated for evaluation.

```
– 2. Evaluation Session Menu –
________________________________

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Remove Evaluation Users
6. List Sessions

0. Exit

Enter your choice : 5

        – 2.5 Remove Evaluation Users –
        ________________________________

        1. List Connection References
        2. Delete Connection References
        3. List Allocation References
        4. Delete Allocation References

        0. Exit

        Enter your choice : █
```

Figure 10–71 :*Remove Evaluation Users Menu.*

The menu shows 4 items.

- **List Connection References** is selected by entering ’1’.This operation displays the applications currently recorded as connected for evaluation purposes.

  An example is given Figure 10–72 There are two evaluation applications currently recorded in the TRDB, TEV_07 and TEV_01.

```
– 2.5 Remove Evaluation Users –
          ________________________________

          1. List Connection References
          2. Delete Connection References
          3. List Allocation References
          4. Delete Allocation References

          0. Exit

          Enter your choice : 1

          2.5.1 List Connection References

          This operation lists the applications referenced
          as connected for evaluation purposes.

          Evaluation_Application
          __________________________________

          TEV_07
          TEV_01
```

Figure 10–72 :*List the connection references of evaluation applications.*

- **Delete Connection References** is selected by entering ’2’.This operation allows to delete the connection reference of an evaluation application that crashed.

  An example is given Figure 10–73. The user is asked for the name of the application whose reference must be deleted. He must also confirm the deletion.

– 2.5 Remove Evaluation Users –

1. List Connection References
2. Delete Connection References
3. List Allocation References
4. Delete Allocation References

0. Exit

Enter your choice : 2

2.5.2 Delete Connection References

This operation deletes the connection reference
for the given application.

Enter the application name to be removed : TEV_07

CAUTION: This operation will delete the connection reference of TEV_07
Do you want to continue (y/n) : █

Figure 10–73 :*Delete the connection references of an evaluation application.*

- **List Allocation References** is selected by entering ’3’.This operation displays the allocations of sessions currently recorded in the TRDB for all the evaluation users.

  An example is given Figure 10–74 The information provided is the evaluation user name, the sessions that he has allocated and their type (EX for EXecution and EV for EValuation).

– 2.5 Remove Evaluation Users –

1. List Connection References
2. Delete Connection References
3. List Allocation References
4. Delete Allocation References

0. Exit

Enter your choice : 3

2.5.3 List Allocation References

This operation lists the session allocations
currently recorded in the TRDB.

Evaluation_User SESSION_NAME TY

rt_user3 EXEC_BEN_002
EX
rt_user3 EXEC_BEN_011
EX
rt_user3 EXEC_BEN_006
EX
rt_user2 EVAL_001 EV
rt_user1 EVAL_001 EV

Figure 10–74 :*List the allocation references of all evaluation users.*

- **Delete Allocation References** is selected by entering ’4’.This operation allows to delete the allocation references of a given evaluation user.

  An example is given Figure 10–75. The user is asked for the name of the user whose allocations must be deleted. He must also confirm the deletion.

```
– 2.5 Remove Evaluation Users –
________________________________

1. List Connection References
2. Delete Connection References
3. List Allocation References
4. Delete Allocation References

0. Exit

Enter your choice : 4

2.5.4 Delete Allocation References

Enter the User whose allocations must be removed : test_user23

CAUTION: This operation will delete the allocation reference of  test_user23
Do you want to continue (y/n) : █
```

Figure 10–75 :*Delete the allocations of an evaluation user.*

10.3.1.2.6 List Evaluation Sessions

If the selection in the ’Evaluation Session Menu’ is ’6’, the ’List Sessions’ operation is executed.

This operation allows the user to list all the evaluation sessions recorded in the TRDB. The data displayed contain the session name, the session state, the session owner and the session creator (see Figure 10–76).

– 2. Evaluation Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Remove Evaluation Users
6. List Sessions

0. Exit

Enter your choice : 6

2.5 List Sessions

This operation lists all the evaluation sessions
recorded in the TRDB. The data displayed is :
the session name, the session state (CREAted, ARCHived, ONLIne, To Be ARchived, To Be EXported, To Be IMported, To Be Completely Deleted, To Be Partially Deleted),
the session owner and the session creator.

NAME	STAT	OWNER	CREATOR
EVAL_BEN_001	ONLI	rt_user3	rt_user1
EVAL_BEN_003	CREA	rt_user3	rt_user3

Figure 10–76 :*Evaluation Session List.*

10.3.1.3 DBS Error Number (DBS_ERR_xxx)

Select the ’DBS Error Number’ operation by entering ’3’ in the Main Menu. This operation provides some information on the high level error messages issued by central DBS. The high level messages sent by Central DBS contain an identifier. By entering this identifier, the user can obtain further explanations on the consequences and the possible recovery actions following the problem detected. Two examples are given at the

Figure 10–77 It shows also the DBS error message sent to the CGSI for the first example.

– DBS RECOVERY SCRIPTS MAIN MENU –

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_ERR_xxx)

0. Exit
Enter your choice : 3

3. DBS Error Number (DBS_ERR_xxx)

Enter the DBS Error Number that you want to be explained (3 digits): 311
<DBS_ERR_311>
Storage of an Event file into the TRDB failed.
Central DBS has not been able to create an Oracle reference for this Event file or it failed to insert the file content into the Oracle event table.
The temporary file should be deleted and the Oracle table references updated, using the DBS Recovery Scripts <File Storage Failure> Menu.

– DBS RECOVERY SCRIPTS MAIN MENU –

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_E

Error message sent by DBS to CGSI:
DBS_ERR_311: Local Event file not stored into the TRDB [Recovery 1.4]
<file name>, <session name>.

0. Exit
Enter your choice : 3

3. DBS Error Number (DBS_ERR_xxx)

Enter the DBS Error Number that you want to be explained (3 digits): 213
<DBS_ERR_213>
The Import of a Test Execution Session from FA medium failed.
The Import of the Oracle Table containing Eng. Value file references for the given Test Execution Session failed because the file where this table has been exported has a wrong format.
Data and references are left on the TRDB disk.
The imported session (NEW_NAME) should be deleted by the the Recovery Scripts <Session is Used> Menu.

Figure 10–77 :*DBS Error Number Explanations*

If the DBS error number selected is out of range, an error message is printed as shown on the Figure 10–78

– DBS RECOVERY SCRIPTS MAIN MENU –

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_ERR_xxx)

0. Exit
Enter your choice : 3

3. DBS Error Number (DBS_ERR_xxx)

Enter the DBS Error Number that you want to be explained (3 digits): 145
Invalid DBS Error Number.

– DBS RECOVERY SCRIPTS MAIN MENU –

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_ERR_xxx)

0. Exit
Enter your choice : 3

3. DBS Error Number (DBS_ERR_xxx)

Enter the DBS Error Number that you want to be explained (3 digits): 938
Invalid DBS Error Number.

Figure 10–78 :*DBS Error Number Explanations*

11 CGS ADMINISTRATION

CGS Administration consists mainly of the following tasks:

- System Administration:
  Adding and removing workstations, test nodes and simulation nodes to a system
- User Administration:
  Adding and removing users from CGS or modifying the user privileges
- Defining the system topology and adapt the CGS configuration parameters
- Maintaining the Storage Resources
- Exporting and importing MDB data and TRDB data
- Startup/ Shutdown of the system and monitoring the system behaviour

11.1 System Administration

11.1.1 Turn On/Boot CGS Hardware

To turn on/boot the CGS hardware it is recommended that the database server host is turned on/booted first. When the boot sequence has finished (login prompt/window visible), the workstations, test nodes and simulation nodes can be turned on/booted in arbitrary order.

11.1.2 Add Additional Workstation Client

Refer to the CGS Installation Manual

11.1.3 Deinstall Workstation Client

Refer to the CGS Installation Manual

11.1.4 Add a Force Simulation Node

Refer to the CGS Installation Manual

11.1.5 Add HP Test Node

Refer to the CGS Installation Manual

11.1.6 Add SUN Test Node

TBS

11.1.7 Deinstall HP Test Node

Refer to the CGS Installation Manual

11.1.8 Removing SUN Test Node

TBS

11.2 CGS User Administration

11.2.1 Add CGS User

A new CGS user must be introduced via the global installation script `install_user` as Checkout User.

```
<hostname>:cgsadmin 1: su<Return>
Password: RootPassword<Return>
# $GSAF_HOME/cgs/util/common/install_user <Return>

Enter the login name for the new CGS user [cgsadmin]: <user> <Return>

Shall the user <user> be added as MDA user? (y/n) [Y]:  <Return>

Enter Oracle Password for the new Oracle User: OraclePassword<Return>

Enter Oracle SYSTEM Password: OracleSystemPassword<Return>

Creating Oracle user : <user>
====================

Please indicate which privilege(s) should be granted :
----------------------------------------------------
    CONNECT           (default for new users)  =>  1
    CONNECT, RESOURCE, CREATE SEQUENCE         =>  2

Privilege(s) [1] : <Return>
.
.           (list of tablespaces)
.
Please enter default tablespace name   : TS_MPS<Return>
Please enter temporary tablespace name : TEMP<Return>
.
.           (some messages and user input for logging file here)
.
Enter OWNER NAME of the Oracle MDB account [MPS]    : MDB_Owner <Return>
Enter OWNER PASSWORD of the Oracle MDB account [MPS] : Password <Return>

Enter OWNER NAME of the temporary Oracle MDB account [MPS_EXPORT]:
Temp_MDB_Owner <Return>
Enter OWNER PASSWORD of the temporary Oracle MDB account [MPS_EXPORT] : Pass-
word <Return>

Enter Oracle USER NAME (ops...) of new user :  OPS$<user><Return>

Enter Oracle PASSWORD of new user :  OraclePassword <Return>

Enter privilege CONFIGURATION_MANAGER or NORMAL_USER: NORMAL_USER <Return>
.
.           ( lot of installation messages )
```

```
.
The user is successfully installed.

Shall the user <user> be added as Checkout user? (y/n) [Y]: <Return>
.
.           ( some messages )
.
install_user: Ready
# exit<Return>
```

11.2.2 Deinstall CGS User

Refer to the CGS Installation Manual

11.2.3 Modify User Profiles

The profile of a user can be changed by opening the TSCV tool and select the ’Properties –> User Profile’ menu option (refer to chapter 8.1.2.6.12 of the CGS User Manual)

It allows to define the CGS role for each user:

Conductor: The user is allowed to setup and shutdown the test system. In addition, everything what a Operator can do is allowed for a Conductor.

Operator: The user is allowed to operate the test, i.e. to give HLCL commands and select data for display etc.

Evaluator: The user is only allowed to evaluate test sessions via TEV or to monitor the tests via synoptics. HLCL Commanding is disabled as well as assigning values to Software Variables.

In addition, for each role a specific set of privileges for TRDB operation is defined (refer to chapter TRDB User Privileges)

Furthermore the screen setup can be allocated to a user and user specific commands to the UNIX can be defined (for the HCI menu ”Specific Commands”).

The user’s profile can also be changed by directly editing the file
$CGS_HOME/config/USER_PROFILES

Further setup of the user’s environment might consist in

- modifying the default setup of the user’s Openwindows desktop
- modifying the Openwindows User Menu (file $HOME/.openwindows–menu)
- modifying the Task Selector entries (file: $HOME/.task_list)

- modifying the default setup for the Message Handler
  (calling the Message Handler, setup up the environment and saving it under $HOME/.cgsi/message_handler_properties)

- defining a default screen setup for the Test Execution for all users

  (as cgsadmin user by calling Test_Execution entry from Task Selector's menu, setting up the desired screen and saving it under 'Basic' or by editing the file '$HCI_HOME/data/screen_setup_pool/basic')

11.2.4 TRDB User Privileges

The `file $GSAF_HOME/dbs/config/dbs_privilege_file.def` implements a mapping between the user profile and the TRDB operations which are allowed. A user profile can access three areas of privileges, which can be combined. These three areas are:

- The DBS concept of TRDB_ADMIN : only the TRDB_ADMIN user can perform the operations of archiving, retrieving sessions and data, export and import.
- The DBS concept of TRDB_MANAGER : only the TRDB_MANAGER user can perform all the operations of deletion without having the ownership of the data. Otherwise the operations of deletion require only the ownership of the data, but no specific privileges.
- TRDB_ADMIN and TRDB_MANAGER can open and close execution sessions.

All the others operations not mentioned above are accessible by all the type of user (TRDB_ADMIN, TRDB_MANAGER and TRDB_USER).

The following mapping is recommended :

```
CONDUCT_USER_PROFILE : TRDB_ADMIN,TRDB_MANAGER
OPERATO_USER_PROFILE : TRDB_ADMIN,TRDB_USER
EVALUAT_USER_PROFILE : TRDB_USER
```

The assignment is defined during CGS installation. If necessary, the general assignment of TRDB privileges may be changed to a different setup.

11.2.5 Show Installed Users

The following scripts are available to show the CGS users installed

$CGSI_HOME/util/common/list_users
$CGSI_HOME/util/common/List_Oracle_Users

A list of CGS users installed is also shown when calling the Task_Selector's 'Installed CGS User' menu option.

11.3 Configuration Setup

11.3.1 Modify System Topology Table

Whenever a test node, workstation, simulation node or SAS has been added/removed, the System Topology Table needs to be changed. One way of modifying the System Topology Table has been defined in the CGS

Installation Manual.

Another way of changing the table is provided with TSCV by opening the TSCV tool and select the 'Properties –> System Topology' menu option. Refer to ch. 8.1.2.6.11 of the CGS User Manual

11.3.2 Maintain CGS Configuration Parameter

CGS defines Configuration Parameter in several configuration files. The parameter can be changed by modifying the files with an text editor. For an overview of parameter that may be modified see appendix K.

11.3.3 Install CMAS / SAS Versions

New versions of CMAS or SAS need to be copied to their predefined location under $GSAF_HOME/sas or $GSAF_HOME/cmas. For installation of CMAS the script 'install_cmas' needs to be used.

The Version Id Table should be updated for SAS by calling the script
$CGSI_HOME/util/common/register_installation –i <sas_name> –v <version> –d <install_dir>

11.3.4 Install Patch Tapes / Maintain Version Id Table

The instructions given in the release notes for the resp. patch tape / update–CD should be followed.

11.3.5 Install "Quick" Patches / CGS External Software

The instructions given in the release notes for the resp. softwrae items should be followed.

If not done already by the installation scripts, the CGS Administrator should register all items in the Version ID Table. To allow this, CGS (CGSI) provides an interface to enter or update entries in the Version ID Table. It allows for registering a software item (e.g. version of a software package, a patch or update etc)

The table is located in the $GSAF_HOME file system and is owned by the CGS Administrator user.

A program vit_manager is available under $CGSI_HOME/bin/sun5 which can be called on any SUN node having visibility to $GSAF_HOME as the CGS Administrator user with the following parameters:

Version ID Table Manangement: List or modify the Version ID Table (VIT).

```
vit_manager –help | –list | –upd_item <sw_item> <field> <value> |
            –add_item <sw_item> <version> <host> <path> | –del_item <sw_item> |

            –query {<sw_item>|ALL} [–format format–list]
```

Parameters:
–help
— get help info
–list
— List the contents of the VIT
–upd_item <sw_item> <field> <value>
— update <field> of <sw_item> with <value> . Entry is tametagged with actual time.

–upd_entry <entry–no> <field> <value>
— update <field> of <entry–no> with <value> . Entry is tametagged with actual time.
–add_item <sw_item> <version> <host> <path>
— Add a new entry (at the bottom) to the VIT. Entry is tametagged with actual time.
–del_item <sw_item>
— Delete the entry with name = <sw_item> from the VIT
–query <sw_item>|ALL [–format format–list]
— query the VIT for a given SW item/product or all items
— the output can be formatted according to an optional format–list
— format–list must be a single parameter with a comma
— or blank separated list of format identifier
— legal format identifier are
— product,version,host,date,time,path

where :
<sw_item> : Name of SW item (max. 20 characters)
<field> may be
’SW_ITEM’ : Name of SW item (max. 20 characters)
’VERSION’ : Version of SW item (max. 20 characters)
’HOST’ : Host where SW item is installed (max. 20 characters)
’PATH’ : Directory where SW item is installed (max. 255 characters)
<entry–no>: Number of entry in the VIT Table

11.3.6 Show Installed Software Versions

To list all items installed and registered in the version id table, the following script is available:

$CGSI_HOME/util/common/list_installations

The list can be optained also via the Task Selector’s ”Software Versions” Option

A list of software items together with their version and install date is displayed (Version Id Table). Each item is displayed with
The entry number
The item name
The version
The host where the item was installed (or where the executable is stored)
The date when the item was installed
The path where the directories of the item are located (or where the executable is stored)

Example:

```
3 GWDU  V2.2 On Host: vicos2s Install_Date: 15.12.1995 18:28:45
    Install_Path: /cgs_develadm/GSAF_HOME/GWDU/bin
```

11.4 Configure Printers for CGS

11.4.1 Install Printer as UNIX/Solaris Printer

Refer to Solaris Manuals.
For the installation of remote printers refer also to the CGS Installation Manual

11.4.2 Configure Printers for MDB / DADIMA

refer to DADIMA Administration Manual

11.4.3 Configure Printers for Test Execution / Test Evaluation

As CGS Administrator, modify `$GSAF_HOME/dbs/user_env/dbs_cshrc` as described below:

- The environment variable VICOS_PRINT_SERVER shall contain the hostname of the Printer Server
- The environment variables VICOS_LASER_1, VICOS_LASER_2 and FA_SAS_PRINTER shall contain the logical names of the configured printers. The variables can contain the same printer name if only one printer is available.

```
hostname:cgsadmin 2: cd $GSAF_HOME/dbs/user_env <Return>
hostname:cgsadmin 3: vi dbs_cshrc <Return>

setenv VICOS_PRINT_SERVER  Your_Print_Server
.
setenv VICOS_LASER_1 Printer_1
.
setenv VICOS_LASER_2 Printer_2
.
setenv FA_SAS_PRINTER Printer_3
.
            ( Save and Exit vi )
:wq <Return>
```

11.5 Oracle Startup/Shutdown

The Oracle Server Startup / Shutdown procedure is described in the Oracle manuals, and the CGS Administrator should refer to this.

In addition, there is an startup / shutdown script installed by CGS under /etc/init.d/oracle, which is foreseen to be called when the server is booted resp. when it is shutdown, but it may be used also for manual startup/shutdown of the Oracle processes.

To call the shutdown of Oracle, enter as root user:

/etc/init.d/oracle stop

To startup, enter

/etc/init.d/oracle start”

11.6 MDB Administration

The following is a short overview on administration tasks for the MDB. The tasks may be allocated to different users, thus implementing distribution of CGS administration to specific experts.

11.6.1 SID Range Extensions

Refer to SID Range Tool Manual

11.6.2 Table Maintenance via DADIMA

Refer to the DADIMA User and Operations Manual

11.6.3 Grant CM Privileges to a User

To list all MDB users currently installed, the script $MDA_HOME/config/oracle_env/list_mdb_users may be used.

The following is a sample output of this script

```
This procedure lists all users that have access to the MDB.
___________________________________________________________________________

Enter OWNER NAME of the Oracle MDB account [MPS]     :
Enter OWNER PASSWORD of the Oracle MDB account [MPS] :

connect to MDB installation account : Connected.

The temporary account that correspond to [MPS] is:

MPS_EXPORT

The following users are currently installed:

User Name                      Privilege
_____________________________________________________
OPS$CGSADMIN                   CONFIGURATION_MANAGER
OPS$GWDUTEST                   NORMAL_USER
OPS$CGS_1                      CONFIGURATION_MANAGER
OPS$CGS_2                      NORMAL_USER
OPS$SDETEST1                   CONFIGURATION_MANAGER
OPS$CSSPROMA                   NORMAL_USER

10 rows selected.
```

To change the privileges of the user's a deinstallation and re–installation with the new privileges is recommended via $MDA_HOME/config/oracle_env/deinstall_user resp. $MDA_HOME/config/oracle_env/install_user (refer to MDA Administration Manual)

11.7 Maintain Storage Resources

11.7.1 Resource Considerations

When performing a test system setup and/or test session creation, the user has to especially consider the amount of disc space and database table space consumed during the test. This is of special importance because CGS has no chance to continue properly in case the discs are full or the database has no space left. Data has to be throughn away and will then be lost for the user.

- general session information (master archive) for execution and evaluation sessions,
- events generated at test execution,
- engineering value logbooks generated at test execution,
- archive files (raw data) generated at test execution,
- and result files being generated during test evaluation (e.g. event lists, data sets etc.).

Not all data will be physically stored within the ORACLE RDBMS as tables, but within the UNIX filesystem. General session information and events are directly stored as rows within ORACLE tables. All other type of data will be stored as files within the UNIX Filesystem, where the ORACLE RDBMS controls references to these files through its UNIX pathname.

11.7.2 Hard Disc (Magnetic Disc)

As described before, parts of TRDB data for execution and evaluations sessions are stored in the UNIX filesystem under the following location: $DBS_HOME/data. Beneath this directory, a structure is created for:

- a directory for storing EXECUTION session files,
- a directory for storing EVALUATION session files,
- a WORK directory for temporary scratch pad use.

Space under these directories is consumed whenever:

- a test execution session is initialised or a test evaluation session is created,
- raw data files are stored in TRDB,
- engineering result files are stored in TRDB,
- Evaluation result files are stored in TRDB,
- a session is retrieved from the optical disc

Space under EXECUTION and EVALUATION session directories is released whenever

- a test execution session or a test evaluation session is deleted,
- an evaluation result file is deleted,
- a session is archived.

It has to be made sure that there is enough space for these files. In case of space problems, repartioning might be necessary.

Repartioning of the UNIX filesystem requires specific system administration / root priviledges and is not further explained here. Please refer to the SUN SOLARIS documentation.

Other disk space consuming files are
- – message log files und $CGS_HOME/log
- – large models stored in the file system
- – MDB reports
- – Load_Scoe files and associated listings under $MDA_HOME/data
- – MDB Export files and BDE files

11.7.2.1 Monitoring of Disc Space

A system housekeeping variable MD_FREE_SPACE (ID=1011) is available on each test node to obtain the amount of free disc space locally available on the test node. This variable refers to the number of available bytes in the file system holding the directory $TN_HOME/...

In case the system is working normally, archive files will be transfered automatically from $TN_HOME to the file system of the central database server. Thus HK variable 1011 should be fairly constant running CGS. In case the archive files cannot be stored on the central TRDB disc any longer, the value should decrease dramatically by 30 MByte/30 minutes. This is an indication for disc space overflow and normally the test should be stopped immediately since CGS cannot guarantee that no data are lost.

Housekeeping values can easily be monitored by defining a SW variable in MDB for each test node which maps to this HK value and which has appropriate limits and associated actions defined.

11.7.2.2 Delete/Export Test Sessions

CGS Administration should monitor the resources used by the TRDB. A regular check should be made, if test sessions can be deleted or exported.

Deletion of test sessions is provided via the TSCV menu option ”Test Session –> Maintain” and the ”Maintain Test Session” Subwindow. Deletion within the default test session should be taken into account as well. The ”Maintain Test Session” Subwindow provides the respective selection criteria and the ”Delete in Default Session” option.

If a test session is not in state ’open’ or ’closed’ but in an inconsistent or error state, the user via TSCV might only be able to list the session (by deselecting both ”open” and ”closed” for Session Status in the ”Maintain Test Session” window of TSCV), but not to delete them. In this situation, the DBS recovery scripts need to be called

To export a test session to an optical disk, the TEV Export/Import Tool can be used.

If a failure occurs during export, again the DBS Recovery Scripts must be called

11.7.2.3 Cleanup of Disks

When disk space is below a certain amount of free blocks, or on a regular basis, the CGS administrator needs to cleanup the disks.

Depending of the amount of archive files and engineering value log files stored in test sessions, deleting or exporting of test sessions will also free the disks where $GSAF_HOME is located.

To support further cleanup of the disks, , the script

```
$CGS_HOME/bin/common/cleanup
```

is provided. It allows to remove or compress CGS log files and temporary files as well as archive files. It provides for selection of the type of files to be removed/compressed.

Before calling the script, the CGS Administrator must verify, that no CGS Processes are running in the system.

The following is a sample output of the script.

```
----------------------------------------------------------------------
-- Cleaning up CGS related files ...
-- Type s or RETURN to skip, r for removal or c for compression !
----------------------------------------------------------------------
-- /gsaf
                                              "cgs/data/log/*.log" [s] r
                                            "cgs/data/log/*.log.Z" [s] r
                                              "cgs/data/log/*.ack" [s] r
                                            "cgs/data/log/*.ack.Z" [s] r
                                                  "mda/data/LOG-*" [s] r
                                       "dbs/data/test/load_scoe.*" [s]
                              "dbs/data/WORK/DEFAULT_TEST_SESSION/*" [s]
                                         "dbs/data/WORK/*/*.EVT.*" [s]
                                         "dbs/data/WORK/*/*.EVL.*" [s]
                                       "tes/data/TES_CONFIG_ERROR" [s]
                          "tes/data/TES_ENVIRONMENT_VARIABLE_ERROR*" [s]
                                       "tes/data/vicos_tes.output" [s]
                                                    "tes/data/AP*" [s]
                                    "tes/data/system_library.hlcl" [s]
                                             "tscv/data/tscv.lock" [s]
                                 "tscv/data/tscv_debug_tscv.trace" [s]
                                                 "tev/data/tmp/*" [s]
                                              "tev/data/tmp_wd/*" [s]
                                 "tss/data/tsp_logfile_from_host*" [s]
----------------------------------------------------------------------
What testnode (one or all) [*]
-- /testnode
                                       "$TN_HOST/archive/TES*.arc" [s]
                                     "$TN_HOST/archive/TES*.arc.Z" [s]
```

```
                                  ”$TN_HOST/replay/TES*.arc” [s]
                                ”$TN_HOST/replay/TES*.arc.Z” [s]
                                   ”$TN_HOST/events/*.EVT.*” [s]
                                      ”$TN_HOST/evl/*.EVL.*” [s]
                                          ”$TN_HOST/*.EVT.*” [s]
                                          ”$TN_HOST/*.EVL.*” [s]
----------------------------------------------------------------------
-- /tmp
                                                   ”adatmp*” [s] r
                                             ”*.sw_versions” [s]
----------------------------------------------------------------------
```

When started with statistic option

```
$CGS_HOME/bin/common/cleanup –statistics
```

information about number of removed and compressed files is provided like

```
.
.
.
   -- /tmp
                                                   ”adatmp*” [s]
                                             ”*.sw_versions” [s]
   ----------------------------------------------------------------------
                Removed: 242
             Compressed: 0
                   Gain: 2358 blocks (a 512 Bytes)
   ----------------------------------------------------------------------
```

11.7.3 Sizes / Estimates of TRDB Data

11.7.3.1 Events

Events logged by an application are first of all written into a pool of buffers. These buffers are cyclically read by an background task within the application, that copies them into a temporary file. After that, this file containing events is sent to central DBS (before loading into the ORACLE RDBMS). The temporary event file is transferred:

- when it reaches a given size (defined by MAX_EVT_NUMBER_IN_LOCAL_FILE),
- periodically (defined by ONL_EVAL_EVT_PERIOD),
- when the session in which the application takes part is closed.

Both parameters have been pre–configured in $GSAF_HOME/dbs/config/dbs_configuration_file.def:

```
! Max Number of Events in a local file (before it is sent to central).
MAX_EVT_NUMBER_IN_LOCAL_FILE 1 450
!
! The period (in seconds) between local EVT file storages on the Central
! DBS server. Then they will be available for online evaluation.
```

```
! CAUTION:
! This value is linked with the DELAY_CLOSE_LOOP value defined in this file.
ONL_EVAL_EVT_PERIOD 1 1
```

Single events are stored as rows within ORACLE tables. The size of a single event can vary, depending on the length of the short and long text field. Both is application dependant. According to own experience, the average size of a single event is 120 Bytes.

The minimum size of a temprorary file is then the size of one event (ie. appr. 120 Bytes). Assuming that MAX_EVT_NUMBER_IN_LOCAL_FILE has been reached, the maximum size of the temporary event file before loading into ORACLE is then 450 * 120 Bytes = 54 KBytes.

11.7.3.2 Engineering Value Logbooks

Engineering values logged by an application are first of all written into a pool of buffers. These buffers are cyclically read by an background task within the application and their content is written into a file. After that, the file containing engineering values is sent to central DBS:

- when it reaches a given size (defined by MAX_EVL_NUMBER_IN_LOCAL_FILE),
- periodically (defined by ONL_EVAL_EVL_PERIOD),
- when the session in which the application takes part is closed.

Both parameters have been pre–configured in $GSAF_HOME/dbs/config/dbs_configuration_file.def:

```
! Maximum Number of Eng. Values in EVL Local Files
MAX_EVL_NUMBER_IN_LOCAL_FILE 1 450
!
! The period (in seconds) between local EVL file storages on the Central
! DBS server. Then they will be available for online evaluation.
! CAUTION:
! This value is linked with the DELAY_CLOSE_LOOP value defined in this file.
ONL_EVAL_EVL_PERIOD 1 1
```

According to own experience, the size of an engineering value can vary between 38 and 547 Bytes.

The minimum size of a file is then the size of one engineering value (i.e. 38 bytes), and the maximum size, as defined by the default configuration value, is 450 * 547 Bytes = 246.15 Kbytes.

11.7.3.3 Archive Files

Archive files stored by TES on the local disc of the HP machines first and transfered into the test result database then lateron.

The size of the archive files depends on the amount of data being archived per second and the time intervall an archive file is open until the next one is created.

Under nominal data rate conditions (80 kbits telemetry, 20 packets per second plus 3 kbits telecommand, 20 packets per second) and assuming the default time period for archiving (30 minutes) the size should be below 30 MByte.

The file system in which the $GSAF_HOME/dbs/data/... directory resides must have sufficient free space to hold the archive files from all test nodes included in a session over the time period planned for the test session and obeying the data rates.

11.7.3.4 Result Files

Evaluation result files will be saved under the users' home directory, in particular under ~/wd/tev/..... Therefore first–of–all the disc quota limitations applied to a user will put constraints. Through the test evaluation functionality of CGS it is possible to directly store a result file into the TRDB. By doing this, the result file will be moved from the users' directory to the location of the TRDB filesystem.

The size of an evaluation result file is non–predictable. It mainly depends on the selection criteria which have been applied (e.g. time frame, number of measurements, ADUs, GDUs selected etc.). Therefore no estimates can be given here.

11.7.4 Monitor and Adapt Tablespaces

11.7.4.1 General

During CGS Operation or on a regular basis, the Oracle Tablespaces for MDB and TRDB need to be monitored, and if required, to be adapted.

An Oracle table is the basic Oracle data storage structure. Oracle tables are held in datafiles called table spaces. Each datafile itself is stored within the UNIX filessystem. Oracle table space size is linked to the size of the tables to be held in the table space. It should be noted that the use of different table spaces minimises data file access contentions and increases efficiency and security.

The table space creation and allocation requires specific priviledges and can only be done by an Oracle DBA, such as the CGS administrator.

During normal use CGS fills as well as frees table space. A table space is consumed whenever CGS creates a table or when Oracle *extends* the table. Oracle extends a table when the current table extent is full. Oracle extends a table automatically without any interaction with CGS.

The datafiles in which the tablespaces are stored are kept within the UNIX filesystem. Whenever the tablespace is altered it has to be made sure that there is enough space for such an extension on the partion of the physical hard disc (see below for resizing of tablespaces). Repartioning of the UNIX filesystem requires specific system administration / root priviledges and is not further explained here. Please refer to the SUN SOLARIS documentation.

The tablespaces can be checked by login as the CGS administrator, calling ’sqlplus’ and entering the command:

select TABLESPACE_NAME, BYTES/1024/1024 from dba_data_files;

Before the administrator can alter, the location of a datafile must be known. The absolute path for the datafile can be retrieved by typing the following syntax within a sqlplus window:

select FILE_NAME from dba_data_files where TABLESPACE_NAME=’<NAME>’;

where *<NAME>* must be specifically set by the administrator for the SQL query:

EXAMPLE:
select FILE_NAME from dba_data_files where TABLESPACE_NAME=’TS_MPS’;

Now an existing tablespace can be resized:

alter database datafile ’<PATH_OF_EXISTING_DATAFILE>’ resize <N>M;

where *<PATH_OF_EXISTING_DATAFILE>* is the FILE_NAME listed in the previous command and *<N>* is the absolute size in Megabytes to be changed to.

EXAMPLE:
alter database datafile ’/oracle_home/oradata/ts_mps.dbf’ resize 170M;

comment : changes the size of the MDB tablespace to 170 Megabytes

As explained above there might be hardware limitations not allowing to resize an existing tablespace. In this case the administrator need to add another datafile:

alter tablespace <NAME> add datafile ’<PATH_OF_NEW_FILE>’ size <N>M;

where *<PATH_OF_NEW_FILE>* can be defined by the administrator and *<N>* is the increment size in MBytes.

EXAMPLE:
alter tablespace ts_mps add datafile /oracle_home/oradata/ts_mps_extra.dbf' size 120M;

comment : adds another 120 Megabytes to the MDB tablespace

11.7.4.2 TRDB Tablespaces

During CGS installation the following Oracle table spaces are created for the TRDB:

Tablespace Name	Size after CGS Installation	Comments
MA_SPACE	Datafile Size: 20 MBytes Initial: 100 Kbytes Next: 100 KBytes MinExtends: 1 MaxExtends: UNLIMITED	This is used for the master archive tables: – General session information – reference information to EVL files, archive files and result files.
MA_INDEX_SPACE	Datafile Size: 1 MByte Initial: 20 Kbytes Next: 10 KBytes MinExtends: 1 MaxExtends: UNLIMITED	This contains cross–reference information to the master archive tables.
ONL_SPACE	Datafile Size: 10 MBytes Initial: 500 Kbytes Next: 500 KBytes MinExtends: 1 MaxExtends: UNLIMITED	This contains reference information for all execution and evaluation data available on line.
MISC_SPACE	Datafile Size: 1 MBytes Initial: 100 Kbytes Next: 100 KBytes MinExtends: 1 MaxExtends: UNLIMITED	This space is used by DBS as a temporary scratch pad area.
EVENT_SPACE	Datafile Size: 100 MBytes (proposed default) Initial: 100 Kbytes Next: 100 KBytes MinExtends: 1 MaxExtends: UNLIMITED	This is used to hold log event data tables for all test sessions. Event tables are created dynamically with the creation of new sessions and their names have the form *EVENT_session_name*. For example : EVENT_DEFAULT_TEST_SESSION, EVENT_TEST_SESSION_1 etc.

The initial tablespace sizes for MA_SPACE, MA_INDEX_SPACE, ONL_SPACE and MISC_SPACE are predefined. For the EVENT_SPACE 100 MBytes is proposed as a default size at CGS installation time, but can be changed by the user when running the installation script.

Note: The most relevant tablespaces for TRDB administration are MA_SPACE and EVENT_SPACE.EVENT_SPACE is the largest of all the Oracle tablespaces and is also the tablespace that gets normally filled the fastest.

During normal CGS use, CGS creates online during test new tables for event data :

- during test execution session initialisation. A table for session event data is created at this stage,
- when a test execution session is retrieved from final archive. Again a table for session event data is created.

CGS table space is freed whenever CGS deletes a table. CGS automatically deletes the event table :

- when a test execution session is deleted,
- when a test execution session is archived.

With CGS 4.2.0 (Oracle 7.3.4.3), the TRDB tablespaces are extended automatically (Extend Size: 1MBytes) when they are filled up.
This avoids situations, where no data can be stored anymore due to reaching the tablespace limit. It leads, however, to situations, where the tablespaces have been extended to high size in previous test sessions, while this is not needed anymore and is wasting disk space. In this situation, manual downsizing by the CGS system administrator is required.

11.7.4.3 MDB and SDE Tablespaces

During CGS installation the following Oracle table spaces are created for the MDB and SDE:

Tablespace Name	Size after CGS Installation	Comments
TS_MPS	Datafile Size: 100 MBytes Initial: 100 Kbytes Next: 100 KBytes MinExtends: 1 MaxExtends: UNLIMITED	This is used for all MDB data tables.
TS_SDE	Datafile Size: 100 MBytes Initial: 100 Kbytes Next: 100 KBytes MinExtends: 1 MaxExtends: UNLIMITED	This is used for all SDE data tables.

MDB and SDE Tablespaces are not selfextending, except the TEMP space.

11.7.4.4 Tablespaces for General Use

During CGS installation the following Oracle table spaces are created for general use:

Tablespace Name	Size after CGS Installation	Comments
RBS_LARGE	Datafile Size: 4 MBytes Initial: 256 Kbytes Next: 128 KBytes MinExtends: 5	This is used for large rollback segments.
TEMP	Datafile Size: 100 MBytes Initial: 100 Kbytes Next: 100 KBytes MinExtends: 1 MaxExtends: UNLIMITED	This is used for storage of temporaty data.

The TEMP Tablespaces is selfextending, except the RBS_LARGE currently not.

11.7.4.5 Monitoring of Tablespaces

Information on the Oracle table spaces can be obtained through CGS HK values TRDB_Event_Usage, TRDB_MA_Usage, TRDB_Misc_Usage (IDs= 1060, 1061, 1062). Their values are displayed in the DB Node Status window of HCI (Online Test Execution)

Housekeeping values can easily be monitored (limit checked) by defining a SW variable in MDB for each test node which maps to this HK value and which has appropriate limits and associated actions defined.

In addition, the total size of a tablespace created at installation in MBytes can be checked by the following command within a sqlplus window:

select TABLESPACE_NAME, BYTES/1024/1024 from dba_data_files;

At any time, the free space of tablespaces in MBytes can be obtained by the following command within a sqlplus window:

select TABLESPACE_NAME, sum(BYTES)/1024/1024 from dba_free_space group by TABLESPACE_NAME;

Mechanisms available to increase TRDB table space are :

- deleting or archiving test sessions. Deleting or archiving a test session will free a fairly significant amount of EVENT_SPACE. Deletion will also free some MA_SPACE but archiving will add to the amount of MA_SPACE used . Deletion and archiving can be performed by a DBS user with the appropriate privileges.
- adding table space. This facility is not available to a normal DBS user. A table space can be added by an Oracle DBA only.

11.7.4.6 Resizing of Tablespaces

In order to extend a tablespace, the CGS administrator has the following possibilities:

- Resize an existing datafile for a tablespace,
- or add a new datafile for a tablespace.

A resizing of the tablespace (if possible) is the most preferred solution. Adding a new datafile should be only done in case of hardware constraints (e.g. not enough free space left on partition etc.).

After applying the change a shutdown and restart of Oracle System is necessary.

Before the administrator can alter, the location of a datafile must be known. The absolute path for the datafile can be retrieved by typing the the following syntax within a sqlplus window:

select FILE_NAME from dba_data_files where TABLESPACE_NAME='<TABLE_SPACE_NAME>';

where *<TABLE_SPACE_NAME>* must be specifically set by the administrator for the SQL query:

EXAMPLE: select FILE_NAME from dba_data_files where TABLESPACE_NAME='EVENT_SPACE';

Now an existing tablespace can be resized:

alter database datafile '<PATH_OF_EXISTING_DATAFILE>' resize <N>M;

where *<PATH_OF_EXISTING_DATAFILE>* is the FILE_NAME of the previous command and *<N>* is the absolute size in MBytes to be changed to.

EXAMPLE: alter database datafile '/gsaf_home/oracle_home/FirstMountPoint/oradata/oracle/event_space.dbf' resize 70M;

As explained above there might be hardware limitations not allowing to resize an existing tablespace. In this case the administrator need to add another datafile:

alter tablespace <TABLE_SPACE_NAME> add datafile '<PATH_OF_NEW_DATAFILE>' size <N>M;

where *<PATH_OF_NEW_DATAFILE>* can be defined by the administrator and *<N>* is the increment size in MBytes.

EXAMPLE: alter tablespace event_space add datafile '/gsaf_home/oracle_home/FirstMountPoint/oradata/oracle/event_space20mextra.dbf' size 20M;

The optical drive is used as an archiving device for archive, export, retrieval and import operations.

Optical discs must be have the Unix file system and should also have been initialised. Instructions to create the file system and to initialise the discs are given in ch. 10.2.6 of this manual. The exact free size of an optical disc depends on the formatting, in particular on the root filesystem created. The average free size after formatting varies between 520MBytes and 580 MBytes for a 650MByte disc.

Theoratically there is no limitation for CGS archiving operations. A record of all data transferred to the optical disc is maintained in the TRDB master archive tables.Whenever an optical disc is full, the FA SAS will automatically ask for a new disc to be inserted.

11.8 Monitor System Behaviour

11.8.1 Monitor Process Status

To monitor the UNIX process status on CGS nodes, the normal UNIX mechanisms can be used.

Furthermore, CGS provides some predefined tools to get the status listed:

a) SUN nodes

Login to each SUN node as cgsadmin user, set the DISPLAY variable to your workstation and call the CGS Task Selector via

```
$CGSI_HOME/bin/sun5/Start.Task_Selector
```

The Task Selector menu should be displayed to you. ”Select Task” –> ”CGS Process Status” gives a list of all CGS related active processes on that node.

Sample Output (for a DB Server node):

```
   98 /usr/lib/netsvc/yp/ypbind
   95 /usr/lib/netsvc/yp/ypserv
  106 /usr/lib/netsvc/yp/ypxfrd
  111 /usr/lib/netsvc/yp/rpc.yppasswdd
15219 vicos_tss_tsp_sun5
15307 dbs_central_eval
15258 dbs_central_arch
15335 dbs_central_exec
 1188 rpc.ttdbserverd
27930 ora_lgwr_oracle
 8690 oracleoracle
 6154 oracleoracle
27926 ora_pmon_oracle
  427 /gsaf_home/oracle_home/app/oracle/product/7.3.2/bin/tnslsnr
 7026 oracleoracle
15433 oracleoracle
15419 oracleoracle
27932 ora_smon_oracle
15302 oracleoracle
28224 /gsaf_home/oracle_home/app/oracle/product/7.3.2/bin/maid.SunOS5.4.release
28223 /gsaf_home/oracle_home/app/oracle/product/7.3.2/bin/oraweb
 6144 oracleoracle
29215 oracleoracle
 8683 oracleoracle
27928 ora_dbwr_oracle
```

The number preceding the process name denotes the PID.

Further information can be obtained for each process via

”ps –f –p <PID>”

in the UNIX command window. Refer to Solaris Manuals for the output format or type ”man ps”.

b) <u>HP Nodes</u>

For HP Nodes, login as cgsadmin user to the node.

Enter

ps –u <cgsadmin user>

to get a list of all cgs releated processes on HP nodes

11.8.2 Monitor Memory Status

a) <u>SUN nodes</u>

Login to the SUN node as cgsadmin user, set the DISPLAY variable to your workstation and call the CGS Task Selector via

$CGSI_HOME/bin/sun5/Start.Task_Selector

The Task Selector menu should be displayed to you. ”Select Task” –> ”Memory Status” gives a window with information on the memory used on that node:

Sample Output (for a DB Server node):

```
Memory usage for host   cgs–test    at time     Thu May 14 11:32:09 MET DST 1998
–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
total:116816k bytes allocated + 31272k reserved = 148088k used, 400056k available
```

The following values are given:
n k bytes allocated : the number of KBytes allocated to the active processes
n k reserved : the number of KBytes reserved by the active processes
= n k used : sum of the previous
n k available : KBytes available for processes. This is the remaining of the swap space available.
for the host where the task selector was started (i.e. normally the workstation):

b) <u>HP nodes</u>

Login to the HP node as root user and call the command ’swapinfo’ (refer to HP UX documentation)

11.8.3 Monitor Time Synchronisation Status

Login to a SUN node as cgsadmin user, set the DISPLAY variable to your workstation and call the CGS Task Selector via

```
$CGSI_HOME/bin/sun5/Start.Task_Selector
```

The Task Selector menu should be displayed to you. ”Select Task” –> ”Time Protocol” gives a window with access to the ntpq tool, which allows to get information on time synchronisation between the CGS nodes.

The ntpq program is activated which allows for the following commands:

```
addvars         associations   authenticate   cl             clearvars
clocklist       clockvar       cooked         cv             debug
delay           help           host           hostnames      keyid
lassociations   lopeers        lpassociations lpeers         mreadlist
mreadvar        mrl            mrv            ntpversion     opeers
passociations   passwd         peers          poll           pstatus
quit            raw            readlist       readvar        rl
rmvars          rv             showvars       timeout        version
writelist       writevar
```

peers

Obtains a list of in–spec peers of the server, along with a summary of each peer’s state. Summary information includes the address of the remote peer, the reference ID (0.0.0.0 if the refID is unknown), the stratum of the remote peer, the polling interval, in seconds, the reachability register, in octal, and the current estimated delay, offset and disper– sion of the peer, all in seconds. In addition, the charac– ter in the left margin indicates the fate of this peer in the clock selection algorithm. Characters only appear beside peers which were included in the final stage of the clock selection algorithm. A ”.” indicates that this peer was cast off in the falseticker detection, while a ”+” indi– cates that the peer made it through. A ”*” denotes the peer the server is currently synchronizing with. Note that since the peers command depends on the ability to parse the values in the responses it gets it may fail to work from time to time with servers which poorly control the data formats.

A ACRONYMS

A

AP	*Automated Procedure*
API	*Application Program Interface*
APM	*Attached Pressurized Module*

B

BDE	*Batch Data Entry*

C

CCU	*Configuration Control Unit*
CDU	*Configuration Data Unit*
CGS	*Columbus Ground Software (or System)*
CGSI	*Columbus Ground System Infrastructure*
CI	*Configuration Item*
CLS	*Columbus Language System*
CM	*Configuration Management*
CSS	*Core Simulation Software*

D

DBB	*Data Base Browser*
DBMS	*Data Base Management System*
DBS	*Data Base Services*

E

EGSE	*Electrical Ground Support Equipment*

F

FES	*Front End Software*
FLAP	*Flight AP (Automated Procedure)*
FWDU	*Flight Window Definition Utility*

G

GTAP	*Ground Test AP (Automated Procedure)*
GWDU	*Ground Window Definition Utility*

H

HCI	*Human Computer Interface*
HLCL	*High Level Command Language*
HW	*Hardware*

I

ICP	*Immediate Command Processor*
I/O	*Input / Output*

J

K

L

LAN	*Local Area Network*

M

MBF	*Mission Build Facility*
MDA	*Mission Database Application*
MDB	*Mission Data Base*
MDE	*Model Development Environment*
MOCS	*Model Observation and Control System*
MPS	*Mission Preparation Software*

N

NWSW	*Network Software*

O

OB	*On-Board*
OS	*Operating System*

P

PL	*Payload*

Q

R

S

SAS	*Special Application Software*
SDDF	*Software Design and Development Facility*
SDE	*Software Development Environment*
SID	*Short Identifier*
SITE	*Software Integration and Test Environment*
SMT	*Simulated Mission Time*
SWES	*Software Entity Software*
SWEU	*Software Exchangeable Unit*
SWRU	*Software Replaceable Unit*

T

TBC	*To be confirmed*
TBD	*To be defined*
TBS	*To be supplied*
TC	*Tele Command*
TES	*Test Execution Software*
TEV	*Test Evaluation Software*
TL–UI	*Top Level User Interface*
TM	*Telemetry*
TPS	*Technical Publishing Software (now called Interleaf)*
TSCV	*Test Setup, Configuration and Verification software*
TSS	*Time Synchronization Software*

U

UCL	*User Control Language*

V

VICOS	*Verification, Integration and Checkout Software*

W

WDU	*Window Definition Utility*

X

Y

Z

B DEFINITIONS

A

Abstract Data Type(ADT)
A set of procedures and functions encapsulating a complex data type. Through the procedural access instances of the data type may be created and destroyed and individual components may be accessed for reading and writing.

Access rights
Define what access various users or applications have to objects or entities.

Acquisition Data Unit (ADU)
An ADU is a data unit that is received by VICOS, which contains data (enditem values) to be calibrated and monitored. In general ADUs contain a set of enditem values in raw format. ADUs may be structured (i.e. a list of enditem values) or unstructured (i.e. a byte array where the values are packed, or a TM packet).

Action
In CGS, the term action is used to describe activation of APs, sending of commands(GDUs), generation of messages or enabling of processing as a result of monitoirng exceptions or conditions.

Application
Program or set of programs performing some specialized user–oriented function (as opposed to general–purpose programs like a DBMS, or an Operating system)

Application Home
The directory location assigned to a particular application of CGS, also called Home Directory.

Application Independence
Application independence is the software characteristic that ensures that the software is not dependent on any database system, microcode, computer architecture or algorithms.

Archive
Refers to the process of relegating obsolete data to external backing storage. The reverse operation (copying archived data back to active storage) is known as restore.
In CGS, either MDB contents may be archived, or complete test sessions.
In VICOS, archiving in addition means storage of data in raw format. All data received or generated by a testnode is archived in OS files (archive files). The files are managed after closure as part of the Test Result DB and made accessible by Test Evaluation S/W.
In VICOS, data storage to magneto–optical disks is called ”Final Archive”. Refer to ”Final Archive”

Atomic Function Block (AFB)
To specify a model in CSS, a simple block having inputs and ouputs defined is used as one of the basic definitions. Such an item is called Atomic Function Block and can be seen as mapped to one procedure of the implementation language AIL.
Two kinds of AFBs are defined: Asynchronous AFBs, which are activated by events, and synchrounous AFBs, which are activated periodically in each simulated time frame.

Atomic Implementation Language (AIL)
Language used for the implementation of atomic function blocks in a model definition.

Automated Procedure A program written in the User Control Language (UCL)**.**
It is a ”compiled program in UCL” which correspond to programs in the usual sense, i.e. which may be not imported by other modules but which may be executed directly.

B

Baseline A set of explicitly defined document issue/revisions, CI constituent versions and lower level CI instantiation baselines, which is used for a CI instantiation.

Base Function The basic function underlying a function block, which is used in associated with the function mask to provide the function block, c.f. Base Function and Function Mask. E.g. An analogy is the invocation of a C compiler with a set of options; the *function block* is the C compiler invoked by ”cc”, the *function mask* is the specified set of options ”–O”, ”–target”, etc. which specify parameters particular to the invocation (”optimisation”, ”cross–compilation”, etc.).

Batch Mode A program operationg in batch mode executes all its actions wothout intervention by a human operator.
In CGS, the programs TEV and TSCV can be operated in batch mode, in addition to the normal interactive mode.
Batch Operations are also available for MDB data entry/export via the Batch Data Entry (BDE) Tool.

C

Calibration Calibration means calculation of values for enditems according to predefined calibration curves, from raw format (i.e. format as received resp. acquired via devices) into calibrated format. The result of a calibrated value is the Engineering Value, which may imply an Engineering Unit to be applied.
CGS extends this definition by referring to calibration also in cases, where a string is extracted from raw data and where state codes are generated for raw discrete values.
Decalibration is the reverse process (–> Decalibration)

CDU domain is a set of MDB item types. CDUs have a specific subset of all MDB item types defined, to allow for working in a specific context of data definitions.

CGS Administrator The user administrating the CGS Installation and the Setup of the system. Its a specific user, intalled as CGS User, with specifc writes. Owns all of the CGS Software executables.
Is allowed to startup and shutdown the system and to define new users.

CGS Server Node in the network that provides file server functions and executes the Oracle Services. In a standard setup, this node also exutes the DB Server node functions, i.e. the DBS Central Processes.

Child In a hierarchical structure, denotes an immediate descendant of a given component. A child is thus located one hierarchical level below its **parent.**

Client A process or program that makes use of services provided by another process or program (called service provider or **server**).

Command Elementary instruction or statement from which a functionality is called.

Commonality Commonality is the software characteristic that ensures the use of interface standards for protocols, routines, and data representations.

Communication Session A communication session is a logical exchange of messages between two users of the network software, such as in a response/request scenario when a request from user A solicits one or more responses from user B. The session can be considered to begin when the request is sent and finished when the last response has been received. This definition of a session is only valid when talking in terms of the network software. It should not be confused with an OSI session because OSI has its own widely accepted definition.

Compilation Unit Smallest unit of code that is accepted by the compiler. In UCL, there are 3 types of Compilation Units: Automated Procedure (AP), Library Specification, and Library Implementation (or Library body).

Component Component is a generic term used to cover any item in the higher levels of the software architecture (i.e. product, assembly and subsystem).

Composite Function Block In CSS, atomic function blocks may be composed together with parameter blocks and other composite fucntion blocks to a new unit, called composite function block.

Condition In CGS, Condition has a specific meaning: It is an attribute of a measurement/software variable or derived value. It specifies, when another measurement is enabled for processing, an AP is started or a limit set is switched, in terms of a specific value of the item it is defined for.

Configuration Control Unit (CCU)

A CCU is a Configuration Unit used to define and control other Configuration Units in the DB. It identifies which specific combination of CDU instances make up a particular configuration. A CCU may, in turn, point to lower level control units, thus leading to an hierarchical configuration tree whose topmost (root) component corresponds to the overall Element Configuration. CCU can be described as directories of items (CDU) in the DB.

Configuration Data Unit (CDU)

CDUs are composite entities containing the actual data items

(grouped into individual units for configuration management purposes). CDUs contain the actual data items, while CCUs are collections of references to CDUs only.

Configuration File A disc file which a software component reads during process start up to retrieve configurable parameter. A configuration file is an ASCII data file and may be changed via a standard editor.

Configuration Unit (CU) Collection of MDB items treated as a single unit for configuration management purposes.
CUs are of two kinds:
(a) Configuration **Data** Units (CDU), which contain the actual data
(b) Configuration **Control** Units (CCU), which contain reference information (CU name, version number, etc.) about other CUs, just like a directory in a file system.

Configuration Data Item All Onboard Data shall be constructed as CDI's.

Configuration Item (CI) An item, which defines a software configuration, that for the purposes of configuration management is required to be considered as a single entity.

Configuration Management The control and coordination of the development of a system.

Consistency Consistency is the software characteristic that ensures uniform design and implementation techniques and notations. In the context of the mission dabase, consistency means valid data with defined references and allowed values.

Consistency state CDUs or CCUs have a defined consistency state:
LOCAL VALID : for a CDU: The data is consistent within the CDU scope
LOCAL INVALID: for a CDU: The data is not consistent within the CDU scope, but may be consistent wthin a CCU scope.
GLOBAL VALID: the data is consistent within a CCU scope.

Correctness Correctness is the degree to which the software component satisfies the specified requirements.

Crew Procedure Language (CPL)
In COLUMBUS, this a special language used onboard for secification of crew actions

D

DATA_API A view based interface to the Mission Database. Provides views for retrieval from the Mission Database, functions for setting the configuration and functions for commit/rollback within the chosen configuration.

Database A common or integrated collection of interrelated data whose purpose is to serve one or more applications.

Database Integrity Refers to the state in which the database is considered to be undamaged (both physically and logically).

Database Management System The software responsible for the actual definition, storage and manipulation of data in a Database at both the physical and logical level.

Database Administrator (DBA) The person(s) responsible for the operation and maintenance of a DBMS. Specific user in Oracle with special access rights.

Data Dispatch The process of supplying data from one instance of CGS/TES to another instance of CGS/TES or CGS/HCI

Data Entry / Data Maintenance
Generally refers to the process of entering and/or updating data in the database.
In this context, the term "maintain" refers to any operation which alters the state of the Database, i.e. add (insert) new data, modify existing data, or delete data.

Data Interface SAS A special type of special appliation Software that can handle ADUs and GDUs under the control of TES. This type of SAS constitutes the Software interface between CGS and the front end equipments that measure data, acquire telemetry, send stimuly or TC packets to the unit under test.

Data Processing SAS A special type of special appliation Software that performs a special data processing. This type of SAS is not controlled by TES but on its own takes the initiative to read data from TES or send data to TES for further processing.

Data Set TEV generates from archive or from the engieering value log extracted and converted (calibrated) values and stores them into objects called "Data Set". Data Sets can then be converted to Data Listings, Graphs or Statistics or may be further processed by user specific programs (SAS).

DB Server In CGS, a database server node is a logical node which provides services to store and manage access to the Configuration DB as well as to the Test Result DB.
A logical DB Server node has to be mapped to a physical processor where it runs on (DB server processor).
The DB Server node normally is allocated to the CGS Server, which is the network node executing the Oracle System and acts for the other nodes as file server.

Deadlock Situation in which two or more user processes cannot complete their transactions because each process is holding a resource that the other process requires in order to complete.

Decalibration Decalibration means calculation of values for enditems according to predefined calibration curves, from an engieering format into a raw format. The result of a decalibrated value is the Raw Value, which can be sent to devices or can be simulated or is suitable to be stored into CCSDS packets.
Calibration is the reverse process (–> Calibration)

Default A value supplied by the system when a user does not specify a required parameter, qualifier, or attribute.

Derived Value — Derived Values are specific enditems whose values are calculated from other enditem's values. An UCL expression can be used to specify the calculation.

Display (or Screen Display) — In this context, refers to an area on the physical screen surface assigned to applications for the purpose of communicating with the human users. It may comprise one or several individual partitions (windows) each assigned to a different application.

Distributed Database — A collection of databases that can be operated and managed separately and also share information.

E

Enditem — A part of the EGSE or Spacecraft (UUT) that can be addressed by the S/W resp. the user and cannot be broken down into lower level items. Examples are: sensors, actuators, S/W variables, measurements,telecommands, automated procedures.
Enditems are addressed via the Columbus name tree (pathname).
Note: When discussing DB structures, the term enditem is sometimes used to refer to any DB entry, differentiating from 'views' to database tables or from virtual trees.

Enditem Mapping — In the Mission DB, user defined enditems may exist. These enditems may have all new aggregates/attributes or may partly be mapped to existing enditems and thus share their attributes.
The mapped attributes are visible to the user as any new attribute of the enditem.
When handled in CGS, the type of the enditem might be handled as the new, user defined type, or may implicitly be handled as the mapped, original CGS type. In this way, user defined enditems may easily extend the existing enditem types by preserving the existing procedures.

Engineering Unit — The Engineering Unit defines the units of measure for a Engineering Value. Refer also to 'Calibration'

Engineering Value — The Engineering Value is the result of calibrating raw values. Refer to 'Calibration'

Engineering Value Log — In CGS, enditem values may be logged (stored with a time tag) in calibrated format in special logbooks as part of the Test Result DB. The values may be evaluated in an offline session without need for re–calibration.

Environment Variable — A UNIX term describing variables whose values are known to a process and definable via UNIX commands. They are indicated by a "$" prefix.
In CGS, environment variables are used to set debug environments for processes and as a shortcut for disk/file pathnames.

Error Message — The collection of data (textual description, attributes such as reference number, criticality, time of occurrence, etc.) which describes the error which has occurred.

Evaluation Definition

A file containing a definition used by TEV. Following types of evaluation definition are defined:
– Selection Criteria (for logging event evaluation)
– Statistic Definition (for statistical analysis)
– Listing Definition (for listings)
– Graph Definition (for graphical presentation of values)
– Report Definition (for generation of a test report).

Evaluation Result File

A file containing the output of a TEV tool for further processing or for storage.

Evaluation Session

A session in the TRDB that contains results generated during data evaluation, i.e. using TEV
Evaluation Sessions may be further extended by adding additional evaluation results.
Evalution Sessions may be stored online in the TRDB or exported to final archive devices (optical disks)

Event Log

In CGS, events (errors, out–of–limit exceptions, user input etc.) may be logged (stored with a time tag) as part of a test session within the Test Result DB. The events may be evaluated in an offline session by producing a selected list of events, ordered according to time tags.
Events mainly consists of messages generated by CGS due to any kind of activity, including the execution of specific event–generating UCL statements.

Exception

An out–of–limits condition or a status inconsistent with the value obtained by the monitoring function. In both cases, the current operational state is taken into account. An exception may also be derived from other values.

Internally in CGS, the term exceptions is also refering to Ada exceptions, which are defined error states of the software.

Execution Session

A session in the TRDB that contains results generated during online test activities. Execution Sessions are opened at the beginning of a test and closed at the end. All archived and logged data is stored in this session.
Execution Sessions may be evaluated by TEV during or after a test. The evluation results are stored in –> Evaluation Sessions.
Execution Sessions may be stored online in the TRDB or exported to final archive devices (optical disks)

Export

In the MDB context, this term refers to the process of extracting data from a DB and preparing it for inclusion (**import**) into another MDB instance.

In the TRDB context, this term refers to the process of extracting test session data from a TRDB and preparing it for inclusion (**import**) into another TRDB instance.

F

Fault	An accidental condition that causes a functional unit to fail to perform its required functions. A fault if encountered, may cause a failure.
Final Archive	Test Session data (Files, TRDB contents) is transfered to the magneto–opical disks during the ongoing tests. This process is called ”final archiving” and is managed by the Final Archive SAS (FA_SAS).
Flexibility	Flexibility is the extent of effort required to change (modify existing) software to accommodate changes in requirements.
Flight Software	All developed or procured software which will execute in the flight configuration after launch. The applicability of requirements in this document to flight software is normally limited to flight software which forms part of the Space segment development. Sometimes the term onboard software is used to mean the same.
Formal Document	A document that is released as part of a CI instantiation.
Formal Language Specification	Formal language specifications are done in extended Backus–Naur Form (EBNF).
Formal Software	A piece of software that is released as part of a CI instantiation.
Function Block	A component of the Simulator (CSS) functionality which is invoked as a single entity, c.f. Base Function and Function Mask.
Function Mask	A mask which provides a map onto the functions provided by the function block, implying the selection or non–selection of each function, c.f. Base Function and Function Mask.

G

Generation Data Unit (GDU)	An GDU is a data unit that contains data generated by a testnode and sent to an SAS. In general, GDUs contain commands/requests or stimuli (i.e. requests for analog / digital output). GDUs may contain as well a complete TC packet.
Ground Data Item	All Ground Data shall be constructed as GDI’s. See document NO TAG for details.
Ground Software	All software that executes in any ground computer or in the flight configuration computers during pre–launch ground operations.
Ground SWEU	All COLUMBUS ground software is configured into ground SWEUs. Each ground SWEU is any software unit or component which can be replaced as a single item. As such a ground SWEU is primarily a configuration management item.

H

Heterogeneous Environment	Refers to a system comprising different types of processors or operating systems.
Hierarchical Name Tree	see **Name Tree**

High Level Command Language (HLCL)

HLCL comprises a set of commands that can be given either interactively as HLCL command or in a predefined HLCL Sequence. In VICOS the language is used for control of the online test execution as well as for test system setup and test evaluation.

HLCL Command
A HLCL command is any command which can be given interactively by the user. HLCL commands can be given in specific command windows or from Synoptics or from a HLCL command sequence. Some of them are translated to UCL interactive commands and transfered to the UCL Interpreter running on the EGSE test nodes.

Home Directory
The directory location assigned to a particular application of CGS, also called Application Home.

Homogeneous system
Refers to a system in which all processors are of the same type or family (usually from one vendor).

Housekeeping Value
In CGS, some internal values are maintained during chkout operations. The values provide for status information on CGS processes and loaded data. They are referred to as Housekeeping Values.

I

Import
In the MDB context, this term refers to the process of receiving or including data from an external (possibly remote) DB into the local DB.

In the TRDB context, it means retrieving a test session from an external device (magneto optic device) into the TRDB and make it available for evaluation.

Integrity
Integrity is the extent to which the software component controls access to system resources. resources here include database items, functions, and software controlled hardware.

Independence
Independence is the software characteristic that ensures that it the software does not depend on its environment (e.g. the computing system, operating system, utilities, I/O routines, and libraries)

Instance
In CGS, the instance of a process resp. a logical node is the actual process/node running on a physical processor. Instances are named with their logical process name (DBS,HCI,TES,TSCV,CSS,TEV) and an instance suffix ”_01” .. ”_32”. E.g. for the different test nodes within a system, 3 might be defined: TES_01, TES_05 and TES_10. Each instance is running on a different machine (test node). Instance Names are mapped to test node’s pathnames using the EGSE–NODE enditem type in the MDB.
Instance Names are defined withn the System Topology Table, where the mapping to physical processor names (host names) is specified.

Interoperability Interoperability is the extent of effort required to facilitate the interface of one software component with other systems or software components.

J – L

Level Name The name which identifies one node at a particular level in the MDB hierarchy. A long path name is a concatenation of level names.

Library see Symbol Library**,** UCL Library

Limit In CGS, monitoring is driven by limits defined for each enditem. Lower and upper limits can be defined. When CGS detects an enditem value which is ’out–of–limits’, a message (’exception’) is raised if a predefined ’count’ of limit violations is reached. Together with the generation of exceptions, automatic actions (start of AP, generation of stimulli/TC) may be initiated. ’Limits’ for discrete or digiital enditems are referred to as ’Expected Values’
There are two kinds of limits: ’hard (danger) limits’ and ’soft (nominal) limits’. Hard limits cannot be changed online and are checked with higher priority than soft limits. Soft limits can be changed by the test operator in an ongoing test (online).
Each enditem may have a set of soft limits defined.

Local Time (LOT) The time to which the system clocks are set is the LOT. It can be e.g. the official local time, in use at the location of the EGSE’s installation or GMT or UTC,
The time server for LOT will be the time of a dedicated clock (NTP Master). In the EGSE this is normally the MTP system clock or an external clock (Master Time Unit).
In CGS all local clocks are synchronised using the NTP software.

Locking Mutual exclusion mechanism used for controlling concurrent access by multiple users/applications to a shared resource.

Logging See ’Event Log’ and ’Engineering Value Log’

M

Maintainability Maintainability is the extent of effort required to find and fix errors in the software component.

MDB Item, MDB Object These two terms are used interchangeably to denote a uniquely identifiable entity that has been defined in the Mission Database. An MDB Object or Item may be decomposed into lower–level items according to the hierarchical nametree conventions, see **Nametree** below.
An **End–Item** is an MDB item located at the lowest hierarchical level (leaf or terminal node), and hence cannot be further decomposed.

Master Archive Refer to ’Test Result DB’

Master Test Processor (MTP) One of the test nodes in the Test Configuration is playing always the role of themaster: Some general functions are only executed

on this node:
– Fetching Housekeeping Values from general services
– Archiving SMT setup commands
– Executing the Overall Setup AP started by TSCV
etc.

Measurement A measurement is a single (analog or digital) intput from a (measurement) device. A measurement may be received by CGS in a TM packet or any other ADU via SAS.
Note: in CGS, sometimes the term measurement is used for all enditems describing engneering values: Software Variables, Derived Values and the measurements as described above.

Mission The performance of a coherent set of investigation or operations in space to achieve space programme goals. A single mission may require more than one flight, and more than one mission may be accomplished on a single flight.

Mission DB (MDB) The Mission Database contains configuration descriptions for the UUT (Columbus Subsystems and Elements) as well as for the EGSE. It contains all definitions describing the UUT, the EGSE configuration and the test to be executed.
It is the central repository for all HW / SW configuration information about Flight Elements, Payloads and associated Ground Support Equipment.

Model Configuration Model Configuration represents a number of Model Functions complete for conversion into an Executable Model Image indepedent of the level of breakdown.

Model Function Model Function represents one or several functions to be simulated on different levels of decomposition. A top level Model Function will be broken down via several levels of decomposition to the lowest level Model Function presentation containing functional definitions coded in a subset of ADA or represented in decision tables.

Model Image The Model Image represents compiled and linked code and data of a Model Configuration ready being loaded and executed.

Modularity Modularity is the characteristic of software that ensures a highly cohesive component structure with optimum coupling.

Monitoring Refer to –> Limit

N

Nametree Hierarchical (tree) structure within the MDB which portrays the hierarchical decomposition of Flight and Ground Configurations into systems, subsystems, equipment, etc. The topmost node of the nametree (called the root node) might designate the Flight Configuration, whereas terminal nodes (leaf nodes) represent the

items that cannot (or need not) be further decomposed, i.e. the so–called **end–items**.
Each MDB object is thus identifiable by a **pathname** indicating the succession of nodes to be traversed to reach that particular item in the Nametree.

Network Time Protocol (NTP) NTP is a time synchronisation protocol used to minimise the offsets between the system clocks of the computers in a network. In this protocol the NTP–clients periodically request time information from the NTP–server. Based upon this information, the NTP–clients adjust their own clocks in order to get the offset w.r.t. the NTP–server to the minimum.

Network A group of computers (workstations) and/or terminals that are linked together to allow the sharing of resources (data and peripherals).

Network Information System (NIS)
A system for management and setup of several UNIX nodes and user in a network. Part of the Solaris Operating System.

Node Physical Node: Any computer within a network.
Logical Node: Any CGS ’function’ that may be distributed to a physical node

Notice window Open look terminology. A window which pops up to display any kind of message. The user input focus is bounded to the window. The user has to select a button (e.g. ”continue”) to confirm, that the message has bee recognized.

O

Onboard Software All software that executes within a flight configuration during on–orbit operations.

Online Test Control (HCI) This software package resp. process provides the main user interface for online testing. It runs on each workstaitons and must be startet by each user via the task selector.

Operability Operability is the ease by which a person can use a system comprising software and hardware.

Operating System The system software that controls the computer and its parts, performing the basic tasks such as allocating memory, and allowing computer components to communicate.

Operator The person who operates any system comprising software and hardware using the provided computer input devices (e.g. keyboard, pointing device, voice input, pushbuttons, switches etc) and/or computer output devices (e.g. screen, printer, lamps etc).

P

Parent In a hierarchical structure, denotes an immediate ancestor of a given component.

Pathname A pathname identifies in a unique way an enditem. A pathname is structured hierarchically according to the name tree defined in the Mission DB.

Portability Portability is the extent of effort required to transfer the software component from one hardware or software system environment to another.

Procedural Interface A procedural interface is an interface which is implemented by a set of procedures, usually grouped into an Ada package(s). The interface may be included in the interfacing component without concern for the underlying implementation, and so allow development and testing of the underlying component.

Product Tree A tree structure that defines the constituent CI instantiation for a particular development.

Protocol Rules and conventions for organizing data to be sent from one machine to another. The protocol enables the destination machine to recognize that the data is addressed to it, check the data to make sure that it is valid, unpack and decode the data, etc.

Q

– No Definition –

R

Realtime Node Either a Test Node, Master Test Processor or a Main Computer System

Reconfigurability Reconfigurability is the characteristic of software that ensures continuity of system operation when one or more processors, storage units, or communication links fail.

Recovery This is the process where a Database which is damaged (or assumed to be so) is restored to a previous state known to be consistent.
In CGS, recovery scripts are provided that allow for restorage of DB contents, especially for the TRDB.

Reliability Reliability is the extent to which the software component consistently performs the specified functions or any interface requirements.

Replay Session CGS allows to retrieve all data stored in archive files of an execution session and display and process the data in the same way as done during the online session. This mode of operation is called Replay, and the execution session foreseen to be replayed, is referred to as Replay Session.

Report In the context of this document, a report may be defined as any human–readable description of one or more MDB items. It is an as-

sorted collection of information usually presented to the user in form of a table or itemized list (tabular format).
A report's specification contains the instructions for generating the report, e.g. data selection criteria, formatting instructions, and sort order.
On request, a report is generated, i.e. the predefined instructions are executed, and the resulting output routed either to the workstation's screen (on–screen report), to the printer or to a user–selected file.

Resource Any of the component parts of the System, or the facilities that it offers (e.g. power, communication channels, etc.).

Response time For interactive transactions, this refers to the time elapsing between the start event (e.g. pressing the ENTER, ACCEPT, or COMMIT key) and a response event (e.g. the first character of the reply reaching the user's terminal).

Reusability Reusability is the extent of effort required to convert a portion of the software component for use in another application.

S

Safety Safety is the absence of hazardous conditions.

Shell The UNIX Shell is the means by which direct access to the UNIX operating system is enabled. Types of UNIX shell such as 'csh', (C Shell), 'ksh' (Korn Shell) or 'sh' (Bourne Shell) exist within which UNIX commands and software programs executed etc. Software programs can be written in the 'shell' language, which can then be executed within the Shell.

Short Identification (SID) A unique identification value (integer) for an end item in the Columbus Name Tree, which is used internally in CGS to access that item.

Simulated Mission Time (SMT) Mission Time starts counting the moment a spececraft is launched or enabled. In order to perform the required EGSE tests, this time can be simulated, and the possibility is provided to manipulate SMT, i.e. to set, stop and continue SMT.
In CGS, SMT is maintained by the Time Server Process (TSP) and may be controlled via UCL statements. SMT is distributed to all nodes. EGSE may further distribute the SMT to frontends and finally to the units driving the onboard time.

SMT domain A set of netork nodes that share the same SMT. In CGS, several SMT domains may coexist. CGS allows to define up to 5 Test Confugurations in parallel, each having a different SMT domain.

Software Variable An enditem describing an engineering value, which is generated by software within CGS or within APs/SAS. A software variable may be bound to an Housekeeping Value of CGS. It then allows to get read access to these values.

Spawn	Term to describe the initiation of an executable piece of software eg running under the UNIX operating system. It will create a standalone program running as a UNIX process. A process can be spawned from another application or directly from UNIX.
Specific Applic. S/W (SAS)	Specific Application S/W (SAS) is the S/W executed in parallel with CGS/TES on the same H/W. It is written in Ada and executed under the same OS as CGS. It is linked to CGS services using VICOS supplied interface libraries. It serves for data delivery to CGS, reception of commands/stimuli and specific processing of data.
Stimulus	A stimulus is a single (analog or digital) output to a (stimuli) device.
S/W Exchangeable Unit	All COLUMBUS ground software is configured as SWEU's. Each ground SWRU is any software unit or component which can be exchanged as a single item. As such a SWEU is primarily a configuration management item.
S/W Replaceable Unit	All COLUMBUS flight software is configured as a set of SWRU's. Some flight SWRUs can be replaced online. Otheres are replaced by reloading the pocessor. All flight SWRU's are configuration management items in the same way as SWEU's.
Symptom	Indicator of faults or failures.
Synoptic Displays	Synoptic Displays is a CGS window service which allow the human user to display and manipulate Synoptic Picture within windows at the workstation screens.
Synoptic Picture	A Synoptic Picture is a predefined graphical picture that represents the physical devices and subsystems (EGSE or UUT). It contains output elements that are dynamicly animated (to indicate status / values) as well as input elements that allow for commanding the devices/subsystems by direct mouse manipulation.
System Administrator	A person responsible for the operation and maintenance of the **operating system** of a computer. In CGS, the system administrator may or may not be the same person as the **CGS Administrator**.
System Tree	The toplevel tree elements within the name tree are defined within the MDB as the System Tree. It may exists in several versions within the same Mission of the MDB.
System Topology Table	The system topology table contains the mapping ' Logical Node Name (Instance Name) to Physical Host Name' of all possible nodes for an EGSE. The logical node name is used by CGS always, while the underlying services (e.g. UNIX) use the physical host names to identifiy the processors in a network. The physical host names must be defined in the UNIX /etc/hosts file.

T

Telecommand (TC)	The data uplinked from the ground to a spacecraft is known as telecommands. It contains requests to the onboard system or bulk data.

In the Columbus EGSE the data is sent to the Spacecraft via the TM/TC Front End. The protocols used are baselined to be CCSDS path service, and therefore TC data is uplinked in CCSDS packets. In CGS, a TC is always a single request. Bulk data is to be uplinked by S/W outside CGS (SAS) using CGS services. TC and Stimuli are handled in a similar way.TC packets are transfered from CGS to SAS inside a Generation Data Unit (GDU).

Telemetry (TM)

The data downlinked from a spacecraft to the ground is known as telemetry data. It contains status data of the onboard system. In the Columbus EGSE the data is acquired from the Spacecraft via the TM/TC Front End. The protocols used are baselined to be CCSDS path service, and therefore TM data is downlinked in CCSDS packets.
In CGS TM packets and measurements are handled in a similar way. TM packets are transfered to CGS inside an Acquistion Data Unit (ADU).

Test Configuration

A test configuration describes the EGSE processing configuration needed for a (or a set of) specific test(s).
It defines the participating nodes, the allocation of functions/data to nodes as well as the SAS to be used/distributed on that nodes. In CGS, the user may select a test configuration defined in the DB before the EGSE is setup. CGS will automaticly setup the system according to the test configuration selected.

Test Node

In CGS, a test node is a specific logical node which provides the data interfaces to the UUT, either directly or via frontend equipment. Test nodes require data from the UUT, generate data to it, process data (calibration and monitoring) and interprete Automatic Procedures.
A logical test node has to be mapped to a physical processor where it runs on (test processor).

Test Results

In VICOS, the term test results is summarising all data produced during a test session that are foreseen for long term storage or offline evaluation. Test Results comprise at least:
– raw data archive files
– event logging data
– engineering value logging data

– printable reports
– graphical output
– data sets produced for storage of enditem values

Test Result DB

The Test Result DB (TRDB) contains all data (test results) produced in different test sessions, either during test execution or test evaluation It contains an index of all data items (Master Archive), indicating the name, the related time frame and the location of storage for each item.

The Test Result DB is managed by DBS and consists of Oracle Tables as well as of OS files.

The Master Archive is a sort of index that allows to retrieve data stored on the Mass Storage according to criteria like time, test session, evaluation session.

The Mass Storage contains physically all the data: Logbooks, Archive Files, Evaluation Result Files. It is accessed thru the Master Archive. The data will be stored temporarily on magnetic disc and then on several storage media (Long Term Storage Medium e.g. optical disc).

The visible part of the TRDB (”Online Data”) is the part that can be directly accessed at a certain instant of time. That means, it is the Master Archive plus the data of the Mass Storage stored on magnetic disc and on the Long Term Storage Media actually mounted.

Test Session

A test session is a term identifying a sequence of operations and activities within a given time frame, related to a specific set of test objectives, together with the generated test results and a reference to the used configuration. A test session has a unique name, and an index of all results produced in the test session is stored in the Test Session Table within the Master Archive as part of the Test Result DB.
There are two kinds of test sessions:
– the Online Test Session,
  which is produced during an online test
– the Evaluation Test Session,
  which is produced by a specific user when data evaluation
  is performed offline.

Time Services (TSS)

In CGS, the maintenance of the Simualted Mission Time (SMT) is provided by a software product called Time Serives Software (TSS). It establishes a time erver process on each CGS node.TSS includes also the setup scripts for the Network Time Protocol (NTP).

Trace

A trace provides a link between two different stages in the development lifecycle in order to provide the traceability for a development.

Traceability

Traceability is the characteristic that provides a thread of origin and a thread of implementation. The thread of origin (or the reason for existence) is from the implementation to the requirements with respect to the specified development envelope and operational environment. The thread of implementation is in the opposite direction and is used for verification purposes.

Trace Object

A trace object is an object from one stage of the software development that has been linked by a trace to another object at another stage of the development (e.g. requirement, HOOD object, test procedure, etc).

Transaction	Single thread of related activities representing a sequence of operations initiated by the occurrence of a stimulus and ending with the required response.

U

UCL Library	Encapsulation mechanism for UCL data structures and operations. (can be viewed as a collection of functions, procedures, types, etc.) A UCL Library corresponds to a *package* in Ada or a *module* in Modula–2.
Unit Under Test (UUT)	UUT means a spacecraft or a part of a spacecraft where the EGSE is connected to and which is foreseen to be checked out using the EGSE and CGS.
User Control Language (UCL)	Columbus Test and Operations language (used for real–time control & monitoring purposes in both the onboard and ground environment)
User	Throughout this document the term **User** refers to any person using CGS–provided services. Users are grouped into different classes or categories and will be assigned different privileges based on the task they perform.
User Event	A specific log item that may be used to identify points within the test activities where evaluation can refer to in a logical way. User events are the only events that have the log type UEVT. InTEV, a time frame may defined according to user events, which are always selectable to the user. In UCL, a user event is generated when calling the procedure USER_EVENT.
User Profile	A description of access rights and user roles. May be defined and setup via the TSCV program.
User Working Environment	A user working environment is a specific S/W environment, which is appropriate to a human user (e.g. test conductor, test observer) and his task. It enables the user to start automatically the applications with the corresponding windows needed for his work (e.g. creation of a Synoptic Displays window and start of the AP's belonging to it). See also 'S/W Environment (VICOS User)'.

V

Version	In the course of its life cycle, a Configuration Unit (CU) usually undergoes several modifications due to evolving user requirements, design changes, etc. It will thus possibly exist within the MDB in many different forms or instances (CU occurrences) commonly referred to as *versions*, e.g. DMS Version 3.2.1. In the Configuration Management (CM) context, however, the various CU occurrences. are classified according to the types of

changes that have been made. The terms **versions**, **issues**, and **revisions** are then used to differentiate between the following 3 cases:
– Modifications due to *requirements changes* which result in a new **version**
– Modifications due to *design changes* which result in a new **issue**.
– Modifications due to *bug fixes, repairs* or *other corrections* (affecting neither the design nor the requirements) which result in a new **revision**.

(In the above example, the CU Identifier ”DMS Version 3.2.1”, therefore, refers to Version 3, Issue 2, Revision 1 of the DMS)

Version_Id_Table The version_id_table is the storage area outside the configuration data base, where the information is stored, which special application S/W (object and data files) and which CGS version is stored/ installed .

VICOS CGS is mainly divided into the Simulation Services (CSS), the Mission DB (MDB/MDA) and the checkout services (VICOS). The term VICOS is outdated, but sometimes still used to describe the checkout services of CGS.

Virtual node A virtual node is an item of the Mission Database that corresponds to an incomplete pathname, that means the attributes of the item are child items and not real attributes like measurement definition etc.

W

Widget An object providing a user interface abstraction (for example, a Scrollbar widget).

Working Directory A disc space containing the intermediate files for a specific tool and a specific user.

Workstation Node A workstation node is a logical node which provides the interface to the operator and executes operator related functions for test execution monitoring and control, for test setup, for test preparation and for test evaluation. It runs the CGS HCI, the TSCV and the TEV or the USer Interface of the Simulation (MOCS)
A logical workstation node has to be mapped to a physical processor where it runs on (workstation processor).

X – Z

X11 X Window System Version 11 is a library which provides functions to write applications with a graphical user interface running on a network.

C END ITEM TYPES

See chapter 7.6

D CGS ERROR MESSAGES

This chapter provides a collection of all possible error messages from the CGS tools and tips and hints how the user can take remedial actions.
Note that all error report mechanism which do not conform with the error message handling as described in chapter 4.3 will be described here in detail.

D-1 Commercial Tools

D-1.1 The SDE

Refer to SDE user documentation

D-1.2 The ORACLE database

All ORACLE errors are documented in the ORACLE user documentation provided by the vendor.

D-1.2.1 ORACLE on–line help facility

Additionally there is an on–line help facility provided by the ORACLE system. A typical ORACLE error message is shown in Figure 1. The error message starts with a three–letter prefix which identifies the facility. In the example you see that ”ORA” is the facility which reported the error and ”01017” is the error number.

Make sure that the file ..../oracle.../bin/oerr is visible on your workstation.

```
marlies@csf_12: sqlplus

SQL*Plus: Version 3.0.12.4.1 - Production on Thu Jun 29 13:33:38 1995

Copyright (c) Oracle Corporation 1979, 1992.  All rights reserved.

Enter user-name: cssaiv
Enter password:
ERROR: ORA-01017: invalid username/password; logon denied
```

Figure 1 : *Typical ORACLE error message*

To get more information about the error open a window and type **oerr** <facility> <error–number> (see the example Figure 2) An explanatory text will be displayed beside the error number. A description of the cause follows in the next line. At last there are suggested actions to resolve the error condition.
Note that there are two types of suggested actions:

- DBA – actions to be performed by the Data Base Administration
  (user needs special knowledge and/or special privileges)
- User – this is what the ordinary user can do

```
marlies@csf_12: oerr ora 60
00060, 00000, "deadlock detected while waiting for resource"
// *Cause:  Transactions deadlock one another waiting for resources
// *Action: DBA - Look at the trace file to see the transactions and resources
//               involved
//         User - retry if necessary
```

Figure 2 : *ORACLE on–line error help*

Figure 3 shows an error message issued by I_MDB. The text is originated by I_MDB, the ORACLE error number is added, the facility is omitted.

- *To get more information about oracle errors reported by I_MDB use the default facility ”ora” and the error number.*
  Note that the ”–” sign standing before the error number has to be neglected to get the correct error message.

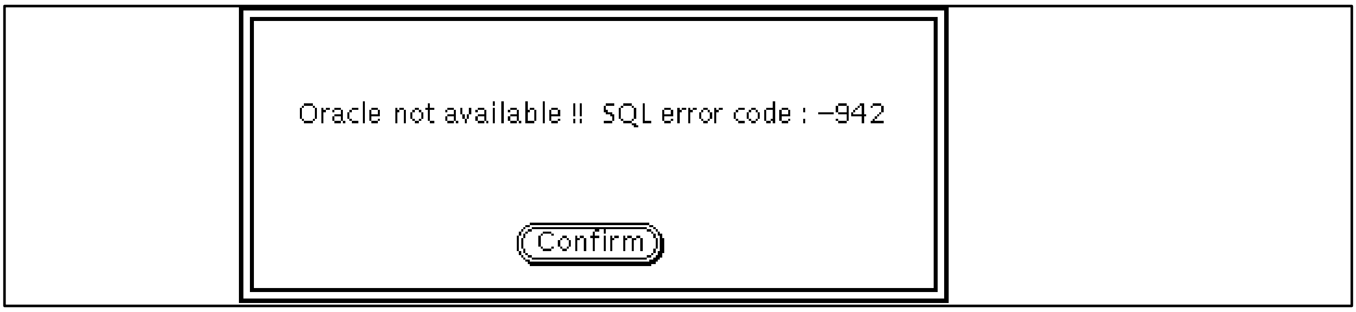


Figure 3 : *Oracle error number added to an I_MDB specific error message*

D-1.3 The ALSYS Ada Compiler

The ALSYS Ada Software Development Environment for HP 9000 gives in the User Guide a description how to read the ALSYS specific Compiler Diagnostics.

Each compilation diagnostic contains three parts: **severity**, **error code**, and **message text**.

There are four levels of **severity** in compiler diagnostics:

LEVEL	DESCRIPTION
Informational	Diagnostics unrelated to warnings or compilation errors. An example is diagnostics regarding the update of an Ada program library.
Warning	Indication by the compiler that a statement might produce unexpected results when that statement is executed under certain circumstances. For example, the compilation of that unrecognized pragma or the compilation of a statement that raises an exception if executed.
Error	Reflects that the compiler detected a syntactic or semantic error.
Fatal Error	An error that is so severe that compilation cannot continue, so compilation of the program is aborted.

The **severity** is indicated by the number of asterisks preceding each of the four diagnostic categories:

- **no** asterisk is displayed for **informational** severity
- **one** asterisk stands for **warning**
- **two** asterisks stand for **error**
- **three** asterisks stand for a **fatal error** that causes the compiler to abort after completing the phase of compilation that detected the error
- **four** asterisks stand for a **fatal error** that results in an immediate abort of the compilation.

A three letter error code shows which part of the compiler issued the diagnostic message. The following table lists the possible error codes.

Code	Title	Description
COD	code generation	Generates machine instructions from intermediate code.
CTX	context manager	Handles **with** and **separate** clauses at the beginning of compilation units.
DAT	data allocation	Determines size and alignment of objects and places into memory areas.

EXP	expander	Produces intermediate code from the source representation. Inserts implied actions, such as checks and calls to tasking support routines.
FIN	final phase	Performs peephole optimization and assembles instructions into bit patterns.
HLO	high level optimizer	Optimizes at Ada source level.
IDE	identification	Identifies the definition of identifiers; performs overloading resolution.
I_O	internal I/O processor	Handles I/O within the compiler.
LAY	data layout	Determines offsets of object within memory areas.
LIB	library manager	Performs all accessing of the program libraries.
LIS	listing processor	Formats source listings and inserts error messages.
LLO	low level optimizer	Optimizes at the intermediate code level.
MAC	manager of all collections	Performs virtual memory management within the compiler.
MAN	compiler manager	This is the main driver routine, which calls all other phases.
OPT	option processor	Decodes compiler invocation arguments.
RAT	rational arithmetic	Rational arithmetic operations.
SEM	semantic analysis	Analyzes the compilation unit according to the semantic rules of Ada.
SYN	syntactic analysis	Analyzes the compilation unit according to the syntactic rules of Ada.
UND	undefined	An error when phase is undefined.

The error message refers to the indicated portion of the program line and is printed directly below the line containing the error.

The error messages issued by the compiler are intended to be clear and self–explanatory. Ada error messages contain as much specific information as possible, including one or more of these components:

– the action taken by the the compiler,

– suggested action to resolve the error condition,

– the rule or language rationale upon which the error depends.

Wherever possible, the messages include citations to specific sections of the Ada Reference Manual, which is abbreviated ”RM” in the compiler message.

incorrect line 29

ALSYS error message

```
Compiled: "MACRO_4\ASYNC"
    Return code: ERROR: Syntax Error
(E) Error Description:
 One error detected. No warnings issued. Compilation failed.
     29              OUTPUT := INPUT ;
                            <-1->
1   **IDE There is a type inconsistency in this expression.

      ======================== More Information ========================
      -> The context requires the following type(s):
         - UNSIGNED_BYTE at line 137 of CSS_TYPES specification, a derived
      type of UNSIGNED_INTEGER8 at line 122 of NUMERIC_TYPES specification
      -> But the expression has the following type(s):
         - UNSIGNED_INTEGER at line 110 of CSS_TYPES specification, a
      derived type of UNSIGNED_INTEGER32 at line 145 of NUMERIC_TYPES
      specification
      ==================================================================

Alsys Ada V5.5.3
Copyright (C) Alsys, 1985, 1991. All rights reserved.
```

Figure 4 : *An example for the ALSYS Ada compiler error messages*

Note that most of the error text was created specific to a certain installation.
The context independent part (ALSYS text) says ’There is a type inconsistency in this expression.” which is easy to understand.
The ” =More Information= ” error report lists the program parts specific to the environment where the error was found. This error report will be created for each error separately and individually.
It is therefore not possible to give further informations about ALSYS error reports.

An example for compiler error messages is given in section D-1.3 .

D-2 Test Preparation

D-2.1 MDA Error Messages

D-2.1.1 Consistency checker error messages

For consistency checker error messages refer to the MDA Reference Manual 4/C

D-2.1.2 Export/import error messages

For export/import error messages refer to the MDA Reference Manual 4/C

D-2.1.3 Batch data entry error messages

For BDE error messages refer to the MDA Reference Manual 4/C

D-2.1.4 I_MDB error messages

For I_MDB error messages refer to the MDA Reference Manual 4/C

D-2.1.5 Generate SCOE Files

For Generate SCOE Files are the same data checks valid as for the consistency checker.

D-2.1.6 GWDU: Ground Synoptic Display Editor

For GWDU error messages refer to the GWDU User Manual (Ref Doc. 2.1.2.1)

D-2.1.7 FWDU: Flight Synoptic Display Editor

For FWDU error messages refer to the FWDU User Manual (Ref Doc. 2.1.2.3)

D-2.1.8 CLS Editor and Compiler

The *Columbus Language System (CLS)* comprises several language related software components for UCL, HLCL and CPL.

Automated Procedures are software programs written in the User Control Language (UCL).

UCL programs are edited and compiled off–line. During the compilation process, the UCL code is transformed into a binary intermediate code which is later executed (*interpreted*) in the target environment by a dedicated program (*interpreter*).

In addition, the compiler generates:

1. a *symbol table* which is internally used by the compiler
2. a *debug table* intended for the source–level debugger
3. an MDB *cross–reference list* intended for the Consistency Checker of the MPS.

In case of UCL programs full checks of availability and compatibility of accessed resources are performed at mission preparation time.

During a UCL editing session errors found by the compiler are marked with a dark rectangle in the editor field.. Additionally the error messages are displayed in a text field in the lower part of the window (see Figure 5). The error messages are written in plain and detailed text. The errors messages are self–explanatory, therefore a list of possible syntax errors is not provided.

Additionally the user gets support from the on–line Syntax Help. The different syntax rules can be checked by pressing the **Help** button (see Figure 6) and the user can copy the desired language construct from the on–line Syntax Help (by pressing the copy button), an easy way to avoid syntax errors.

Internal errors (i.e. errors in the CGS software), which may occur, are clearly marked as internal errors. There are no corrective actions recommended to be performed by the user. The internal error has to be reported to the CGS maintenance team.

If an internal error message is displayed, write a SPR (software problem report) immediately and report the error to the maintenance team.

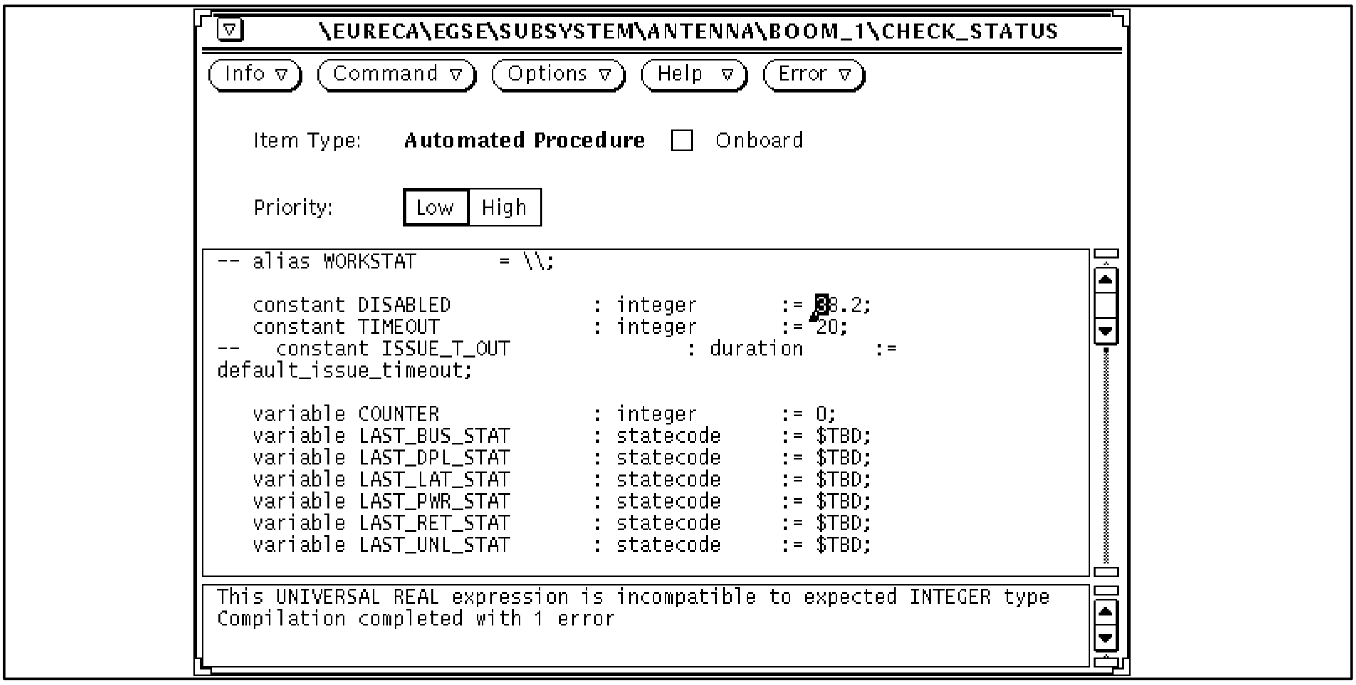


Figure 5 : *The UCL compiler error message is displayed in the lower part of the editor window.*

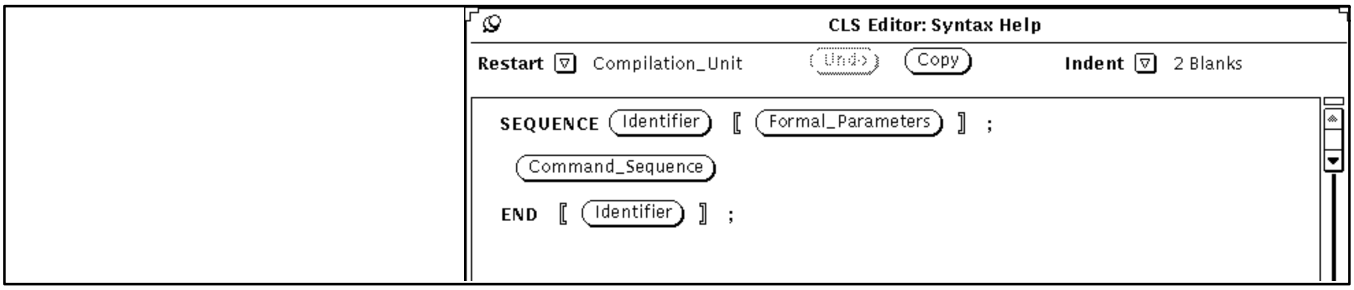


Figure 6 : *The on–line Syntax Help shows the UCL syntax*

D-2.2 HLCL command sequences

HLCL is an interactive command language used in different environments: within an interactive session the user types comands which are executed immediately by the HLCL command interpreter.

HLCL command sequences are edited and compiled off–line. The error handling mechanism is just the same as provided with the UCL editor. Also the Syntax Help function is available for HLCL.

Before the HLCL command sequence can be stored in the database a syntax check will be performed. Errors are reported in the same way as descrbed in section NO TAG (automatec procedures). Figure 7 shows the erronous HLCL sequence.

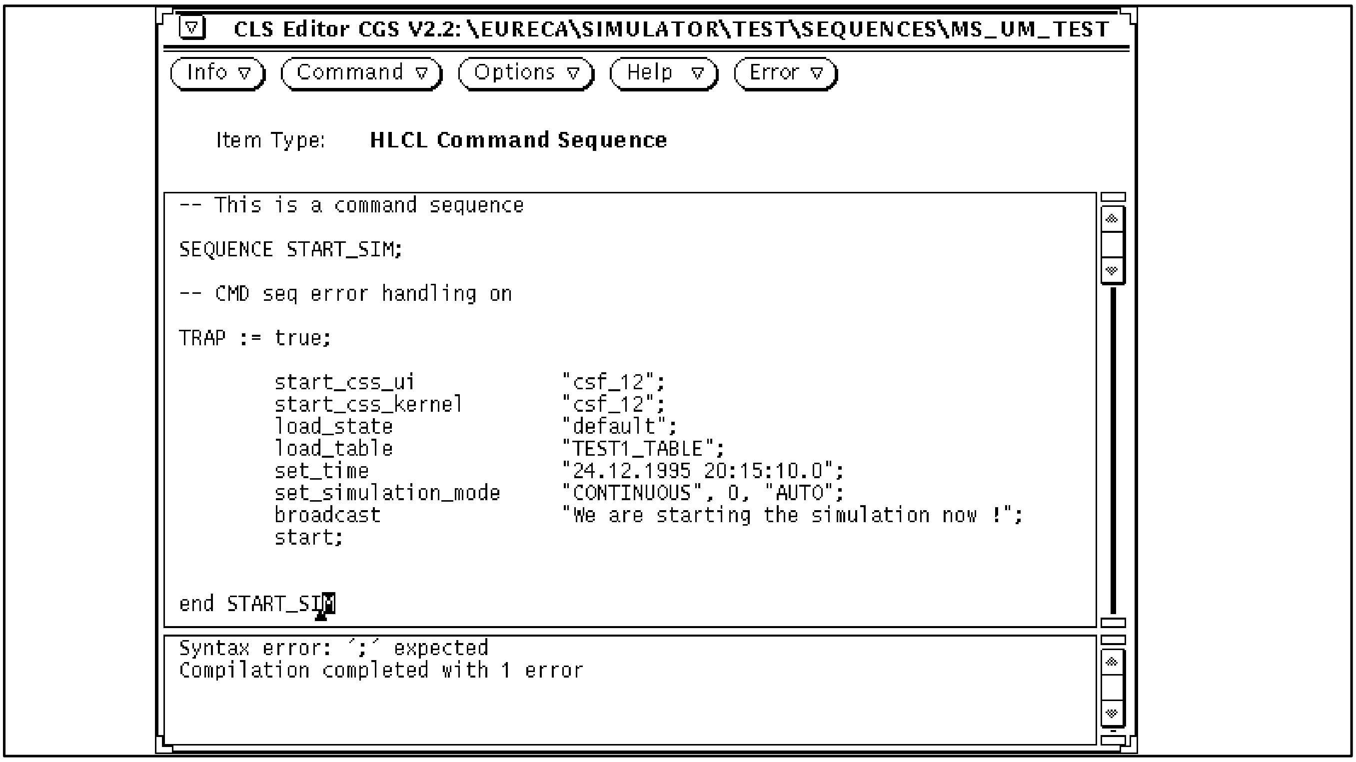


Figure 7 : *A syntax error is indicated by a dark rectangle, the error message is displayed in the lower part of the editor window.*

The syntax of the HLCL language written in a command sequence is *not* completely equal to the way HLCL is used as interactive language.

Within the command sequence the same syntax rules apply as in a session, but some session specific relaxations and extensions are not valid:

- Abbreviations are not allowed.
- The end of a line is not a command terminator, i.e. commands may be formatted over several lines if desired but must be terminated with a semicolon (in any case).
- Engineering units must not be omitted.

D-2.2.1 Model Development

D-2.2.1.1 General

The following section gives a short overview about the different error handling methods used in CSS.

Note that only run time errors which occur during model execution are delivered to the central CGSI error handler.

Depending on the action the user is currently performing (model development, model compilation, model observation and model execution) there are different ways to display errors and warnings.

Note that different parts of CSS use different error handling methods.

D-2.2.1.2 Error messages during model editing

The Composite Editor window has a separate message line at the top of the editor area.
During model editing information and error messages are displayed in the message line.

Not all information displayed in the message line are error messages !

Most error messages give also some advice to take corrective actions (see Figure 8)

While editing a decision table or AIL code error messages are displayed in the message line of the appropriate Atomic Editor subwindow. (see Figure 9)

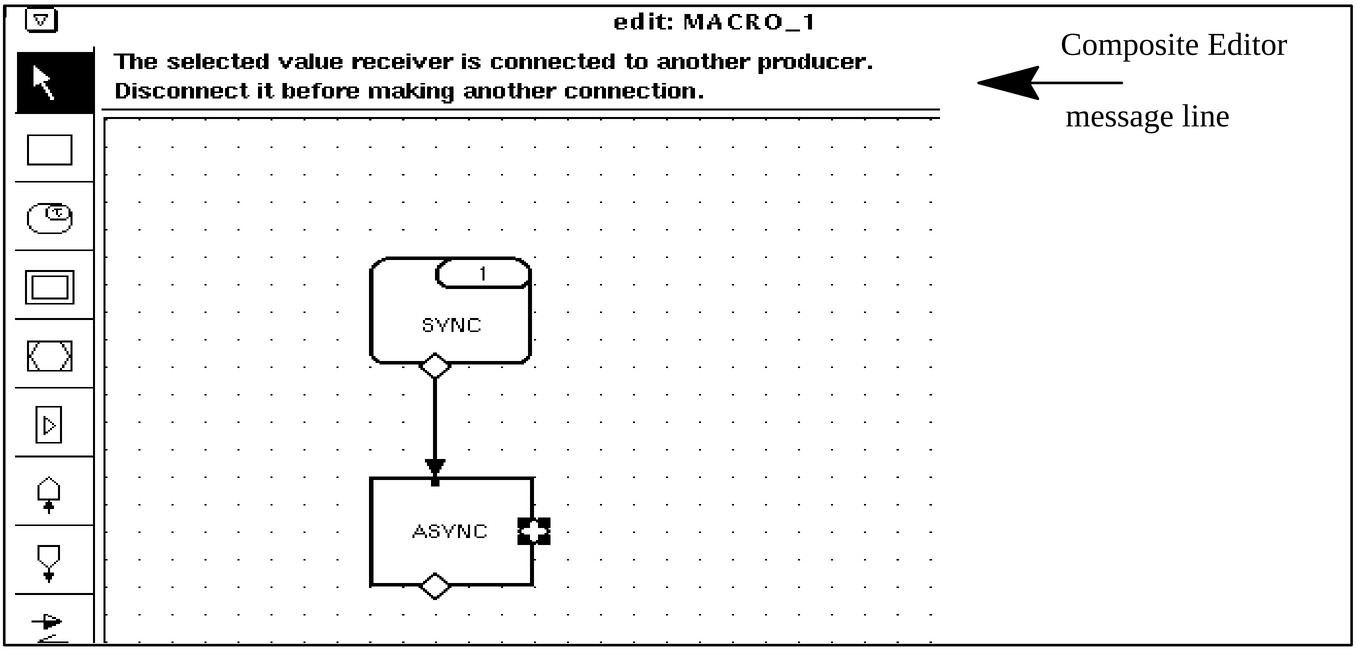


Figure 8 : *Error message displayed in the Composite Editor message line*

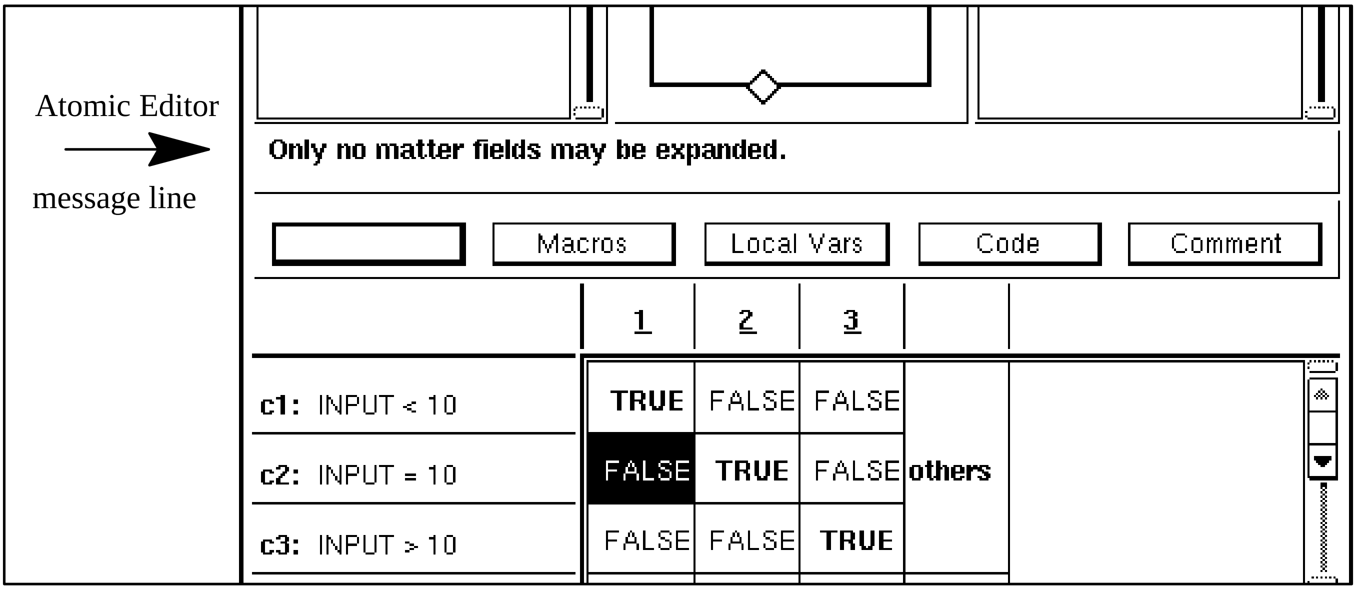


Figure 9 : *Error message displayed in the Atomic Editor message line*

D-2.2.1.3 Error messages during model configuration

Errors and warnings found during the MDE rule check are displayed in an extra scrollable Rule Check error window.
(see Figure 10)

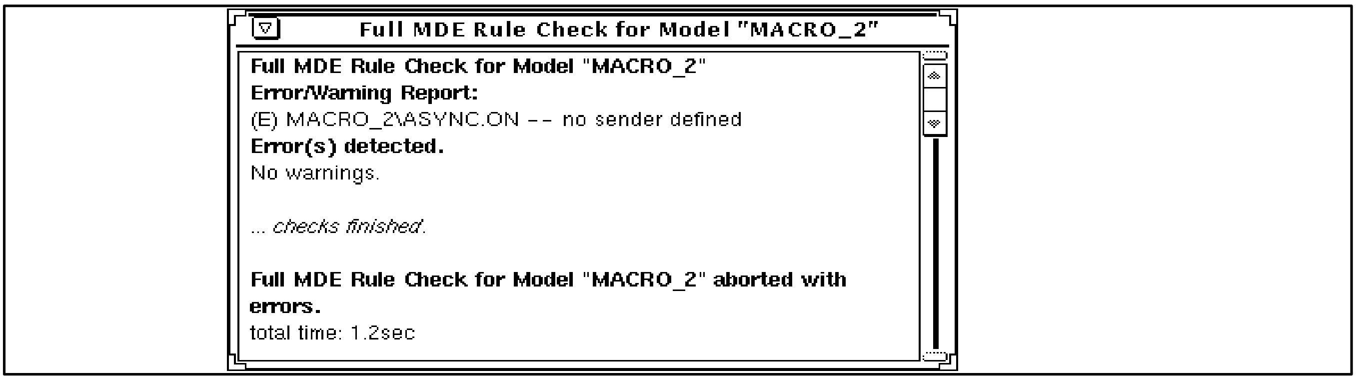


Figure 10 : *The CSS MDE rule check window*

As soon as the Ada compiler starts the transcript window shows the protocol of the proceeding.

The following example gives an impression how to read the transcript window contents.

... compiling model functions ...	*Compilation starts ....*
(E) CTGConnection Protocol Error No.: -40 (Ada Error)	*CSS internal error number*
Error Description: COMPILATION_TERMINATED_WITH_SYNTAX_ERROR	*CSS internal error description*

Figure 11 : *Transcript window contents – First part*

CSS describes the error with an error number and an error description in plain text.

incorrect line 29	`29    OUTPUT := INPUT ;`
ALSYS error message	`1   **IDE There is a type inconsistency in this expression.`

```
Compiled: "MACRO_4\ASYNC"
    Return code: ERROR: Syntax Error
(E) Error Description:
One error detected. No warnings issued. Compilation failed.
     29                OUTPUT := INPUT ;
                              <-1->
1   **IDE There is a type inconsistency in this expression.

      ======================= More Information =======================
      -> The context requires the following type(s):
        - UNSIGNED_BYTE at line 137 of CSS_TYPES specification, a derived
      type of UNSIGNED_INTEGER8 at line 122 of NUMERIC_TYPES specification
      -> But the expression has the following type(s):
        - UNSIGNED_INTEGER at line 110 of CSS_TYPES specification, a
      derived type of UNSIGNED_INTEGER32 at line 145 of NUMERIC_TYPES
      specification
      ================================================================

Alsys Ada V5.5.3
Copyright (C) Alsys, 1985, 1991. All rights reserved.
```

Figure 12 : *Transcript window – Second part*

The incorrect line is displayed with its line number.

The error message refers to the indicated portion of the program line and is printed directly below the line containing the error. The ALSYS compiler error message is built up as follows:

– **1** is the error number (there is only one error detected)

– ****** two asterisks which indicate the severity of the error,

– the error code **IDE** which indicates the part of the compiler which issued the diagnostic message

– and the message text (consist of a description of the compiler error)

The section ”More Information” contains suggested actions to resolve the error condition and/or the rule or language rationale upon which the error depends.

Refer to section D-1.3 for more information about the Alsys Ada Compiler error messages.

In many cases the compiler gives a reference to the related chapter in the **Programming Language Ada Reference Manual** (which is abbreviated ”RM” in the compiler message) with some additional explanatory text. (see Figure 13)

A:error: RM 3.3: base type of expression must be signed_byte, line 118 in css_types from file /usr/columbus/users11/css_test/v3/css/basic/common_sources/source_current/css_types_.a	*reference to chapter 3.3 of the Ada Language Reference Manual*

Figure 13 : *Transcript window contents – Third part*

The protocol closes with CSS internal error messages and the statement that the configuration was aborted.

CSS internal error numbers	(E) CTGConnection Protocol Error No.: -128 (Internal Error; CTG SERVER: Unhandled Exception) (E) CTGConnection Protocol Error No.: -1 (Broken Connection; Connection to CTG Server broken) **Configuration for Model "MACRO_4" on csf_hp2 aborted with errors.** total time: 2min 7.2sec

Figure 14 : *Transcript window contents – Fourth part*

Errors produced during DBB or MOCS operations appear inside a pop–up menu with **OK** button. After pressing the **OK** buttons the CSS session proceeds normally.

D-3 Test Setup and Execution

D-3.1 VICOS Error Messages

In the following section the message types and message groups are shown as used by VICOS. The source where the messages for the groups/types are generated is given to indicate the error origin in general. The general description of 'event' shows the kind of errors/messages generated for the group/type.

Legend: Severity: Severity Level used in the Message Window
ADV = ADVISORY
ORD = ORDINARY
SEV = SEVERE
FAT = FATAL

Group : Identifies logical Group of messsages
Can be used in TEV to select log events

Type: Identifes type of message

MSG/INFO	Information to the user
ERR	Error Message
WRN	Warning Message
EXC	Monitoring Exception (Soft Limits)
ALRM	Monitoring Exception (Hard Limits)

Source: CGS Product / Process delivering the message

Event: Events raising the messages

Group	Severity	Type	Source	Event
SYST	ADV	MSG	VCS	VICOS System Info Messages
	ORD	ERR	VCS	VICOS System Error Messages
	ADV	WRN	VCS	System Warnings
TSCV	ADV	MSG	TSCV	TSCV Internal Event
	ORD	ERR	TSCV	TSCV related Error Messages
	ADV	WRN	TSCV	TSCV related Warnings
TEV	ADV	MSG	TEV	TEV Internal Event
	ORD	ERR	TEV	TEV related Error Messages
	ADV	WRN	TEV	TEV related Warnings
HCI	ADV	MSG	HCI	Messages and Internal Events
	ORD	ERR	HCI/TES	HCI related Error Messages
	ADV	WRN	HCI	HCI related Warnings
HLCL	ADV	MSG	HCI: HLCL Interpreter	HLCL Commands
	ORD	ERR	HCI: HLCL Interpreter	Error Messages on User Input
	ADV	WRN	HCI: HLCL Interpreter	Warnings
TES	ADV	MSG	TES	TES Internal Event
	ORD	ERR	TES	TES related Error Messages
	ADV	WRN	TES	TES related warnings
	ADV	INFO	SAS	Messages received by TES from SAS
MON	SEV	EXC	TES: Monitor	Monitor Exceptions (Soft Limits)
	FAT	ALRM	TES: Monitor	Danger Limit Violations (Alarms)
	ADV	WRN	TES: Monitor	Monitor initiated AP’s/commands
	ADV	INFO	TES:	Monitoring related error
DACQ	ORD	ERR	TES	Acquisition Failures
	ADV	INFO	TES	Acquisition Events
DGEN	ORD	ERR	TES	Generation Failures
	ADV	INFO	TES	Generation Events

Group	Severity	Type	Source	Event
ARCH	ADV	INFO	TES	Events related to archiving
	ORD	ERR	TES	Errors related to archiving
CMD	ADV	MSG	TES:	Commands received from remote nodes / workstations
	ORD	ERR	TES:	Erroneous commands / Errors in command execution
	ADV	WRN	TES:	Warnings given on commands received
DDS	ORD	ERR	TES: DDS Service	Data Delivery related failures
	ADV	INFO	TES: DDS Service	Data Delivery Warnings
UCLI	ORD	ERR	TES: UCL Interpreter	UCL Interpretation Errors
	ADV	INFO	TES: UCL Interpreter	UCL Interpretation Messages
LOG	ADV	MSG	TES: UCL Interpreter	UCL ”LOG” Statement (= User Events)
UEVT	ADV	INFO	TES: UCL Interpreter	UCL ”LOG” Statement (= User Events)
REPL	ADV	INFO	TES	Items as read from archive (Replay Mode)
	ORD	ERR	TES	Errors during Replay Mode
C DB	ORD	ERR	Products calling MDA	CDB(MDB) Access failures
	ADV	WRN	Products calling MDA	Warnings /Events during CDB(MDB) access
	ADV	INFO	TES	Events during CDB(MDB) access
TRDB	ORD	ERR	Other products calling DBS	TRDB Access failures
	ADV	WRN	Other products calling DBS	Warnings /Events during TRDB access
SAS	ORD	ERR	TES	Error in SAS Handling
	ADV	INFO	TES	Advisory Messages in SAS Handling
	ADV	WRN	TES	Warning Messages in SAS Handling
	ADV	<XXX>	SAS	Messages received from SAS via TES_API <XXX> is type specified by SAS: MSG,INFO,WRN
	ORD	ERR	SAS	Messages received from SAS via TES_API
	SEV	EXC	SAS	Soft Monitoring Exceptions received from SAS
	FAT	ALRM	SAS	Hard(Danger)Monitoring Exceptions from SAS
UUT	ADV	INFO	SAS	UUT Messages received via SAS
TST	ADV	MSG	Test Messages	(for debugging)

D-3.2 Test Setup

D-3.2.1 System Setup / Shutdown

stop_cgs (25423) begin.",

Hosts: cgs–test cgs–test1 csf_hp2 cofprime

cgs_startup (24378) end.

cgs_shutdown (23131) begin.",

Hosts: cgs–test cgs–test1 csf_hp2 cofprime

stop_cgs_node (27161) end. Errors !",

Exit code 1
Effective user = cgs_1, NO permission, login as cgsadmin
—> The user cgs_1 has tried to shutdown the system. This is only allowed for
—> user cgsadmin

D-3.2.2 Test Configuration Setup and Verification (TSCV)

D-3.2.2.1 TCSV error conditions

The TSCV product runs on a CGS workstation, and can execute in two modes: either as a batch program which executes its task solely on basis of data provided as parameters on the command line, or an interactive program. It uses interfaces provided by other CGS products, but does not provide any interface to other products.

Interactive mode

All error messages created in WIMP mode will be displayed to the user in a pop–up dialog window and sent to the Message Window.

When TSCV executes in the interactive mode, all error conditions not handled internally will be escalated to the user, and resolved by him. No foreseen error condition shall cause the program to halt.
Some errors may however not be resolved by the user of TSCV, but must be resolved by system maintenance, database maintenance etc.

If TSCV for some reason has not been able to perform user authorization (e.g. no user profile could be found for the current UNIX user account) TSCV will terminate (after prompting the user). The user account must be established – with sufficient authority to run TSCV.

Batch mode

When TSCV starts executing in batch mode runs in batch mode, it will communicate a 'result' as a Unix completion status. It can be read by the program which invoced TSCV to determine the status upon termination. The following exit codes may be produced by TSCV in batch mode:

- Success 99;
- CCU selection invalid for some reason 100;
- invalid mission name 101;
- invalid element configuration 102;
- invalid system tree version 103;
- invalid CCU name 104;
- invalid CCU version 105;
- invalid system tree node name 106;
- can not connect 107;
- invalid test configuration 108;
- EGSE setup failed 109;
- test failed 110;
- could not create test session 111;
- test configuration in use 112

D-3.2.2.2 TCSV error messages in the Console Window

*** ’Alarm’ from TSCV! The program is no more able ***
*** to ’touch’ its lock file to indicate that it is ***
*** alive’. Program exception STORAGE_ERROR occured ***
*** in the program unit called ’ALIVE_TASK’. ***
—>
—> kill process and restart TSCV

Main Program abandoned: Exception WIN_ILLEGAL_HANDLE
—> Check DISPLAY variable: Is is set to ”:0.0” ?
—> Change to the correct value: setenv DISPLAY <host>:0.0

D-3.2.2.3 TCSV error messages to the Message Window

Another TSCV instance is already executing.
In module: TSCV_MAIN.TSCV
Only one instance of TSCV can be active at a time. This instance of TSCV will quit.

Comms error in DBS_COMMS.INIT_LOCAL
CONNECT_TO_VICOS_PRODUCT returns ETIMEDOUT
—> This message appears when TSCV is started and DBS is not running
—> TSCV will then start DBS; so the message can be ignored
—> In other situations:
—> TSCV is not able to connect to DBS;
—> Verify DBS processes are active

Comms error in DBS_COMMS.SEND_WITH_ACKNOWLEDGE_TO_CENTRAL
SEND_MESSAGE returns ETIMEDOUT
Problem with connection to DBS: DBS_COMMUNICATION_PROBLEM Stop and restart TSCV
—> DBS is in a bad shape
—> shutdown dbs (kill processes)
—> restart TSCV (will restart DBS)

Could not convert SID to PATHNAME: Error retrieving MDB data.
—> Was there a previous error indicating access problem to the MDB ?
—> Is there an old testconfiguration used which has no correspondence in the MDB anymore ?

DBS is in Error (pop up window)
—> the overall status of DBS is NOT_OK
—> Is the DISK full ?
—> Remove test sessions from the TRDB
—> cleanup the disk partition where the TRDB is located
—> must have the percentage of the overall space free
—> that is defined in the file $DBS_HOME/config/dbs_configuration.def

DBS error:INVALID_SESSION_NAME&&The error occured when creating session xxxxx
—> The session name given when creating a test session is invalid
—> The session name must be in CAPITAL letters !

Default test session already in use by other.

Only one test configuration may utilize the default test session at a time.

Deleting session:Failed to delete session <session_name>.:SESSION_IS_USED

—> The session is currently in use (open or evaluated)
—> If no user is active: There was probably a crash of a tool when accessing the session
—> Use DBS recover scripts to force the removal of the session:
—> As cgs_administrator execute $DBS_HOME/util/sun4/delete_sessions_like <session_name>

Error commanding TES: Setup failed, error in TES_01

– COMMUNICATION_ERROR in INIT to TES_01
—> INIT command from TSCV to TES failed
—> could be several reasons:
—> a) Connection to MDB failed
—> there was no CCU written by the "Generate SCOE files"
—> function within I_MDB
—> b) Programming error in TES
—> Write SPR resp. use correct TES version
—> c) TSS is not running
—> check if the tss_tsp process is running on the test node
—> Connection to DBS failed
—> check if DBS is running; Can other process connect to it ?

Error commanding TES: Setup failed, error in TES_01

– OTHER_ERROR in INIT to TES_01
—> INIT command from TSCV to TES failed
—> could be several reasons:
—> a) Data Loaded from MDB are erroneous
—> Use MDB consistency checker to verify data loaded from MDB
—> (on CCU as used for the current Test Configuration)
—> b) Programming error in TES
—> Write SPR resp. use correct TES version
—> C) an exception occured while initialising HCI, DBS, TSS or the local data pool within TES

Error executing Set–up: Error when singalling new global CCU to HCI

Error sending message:ETIMEDOUT
—> For each HCI which is in the test configuration, but not running
—> such a message occurs
—> Ignore if HCI is really not running

ERROR in L_CY_EVENT_SESS_SEND.CREATE_LOCAL_FILE

ADT_PACKED_FILE.CREATE fails for file /testnode/cgs–test/TSCV_01.11–02–98_11:34:", E_CREATE_FILE_FAILED. Cause: Cannot create file. Consequence: CREATE_LOCAL_FILE failed. Recovery act: none. Debug info: dbs_local_cyclic_event_sess_sender.a

Error occured when starting TES_01

In module EXTIF_OS.START_APPLICATION Symptom: CONNECTION_TIMEDOUT:
Setup failed: Launching TES_01 Could not start TES_01
—> The Process_Creation_Server process is not running or not
—> answering when TSCV tries to launch a test node (TES_01)

—> Get UNIX node name for TES_01 from
—> $CGS_HOME/config/SYSTEM_TOPOLOGY_TABLE
—> Verify all servers are running. To start Process_Creation_Server:
—> Execute ”$CGSI_HOME/bin/common/Start_Process_Server”

External error when calling MDB to set local default
In module MDB_SESSION.SET_LOCAL_DEFAULT Symptom: ORACLE_ERROR.
—> Is Oracle running ? Check on DB server node if Oracle processes are up.
—> Is the MDB installed correctly ?

EXTIF_DBS.CONNECT_TO_DBS: Error when calling SystemTopologyTable.Who_Am_I:
EXTIF_DBS.CONNECT_TO_DBS: Error when calling
SystemTopologyTable.Who_Am_I:TYPE_UNKNOWN
—> Is TSCV entered on this node in the
—> $CGS_HOME/config/SYSTEM_TOPOLOGY_TABLE ?

Failed to access the Mission Database (MDB).
– EXTIF_MDB.GET_CCU_VERSION_LIST:Operation MDB_SESSION.READ_SIDS raised exception”,
The initiated operation can not be completed because of an internal error in the Mission DB provided procedures. Please contact the system manager”

–EXTIF_MDB.GET_CCU_VERSION_LIST:Operation MDB_SESSION.GET_LIST_OF_TC_IDENTIFIERS
The initiated operation can not be completed because of an internal error in the Mission DB provided procedures. Please contact the system manager”);

Failed to check TSP status on node <unix–node>
EXTIF_TSS.VERIFY_LOCAL_TSP_STATUS:Operation TSS_NODE_STATUS returned ETSUNKNOWNF”,
The inquiry of the Time Server Process status resulted in an error indication.
Unable to open NTP status file!
The SMT services on cgs–test may be incorrect.”);

Failed to clear the SMT time.”,
EXTIF_TSS.VERIFY_LOCAL_TSP_STATUS:Operation TSS_NODE_STATUS returned ETSUNKNOWNF”,
Unable to open NTP status file!
The SMT services of all nodes of the test configuration remains set–up as before.

Failed to close test session.
EXTIF_DBS.CLOSE_SESSION:Operation DBS_RPI.CLOSE_EXECUTION_SESSION returned COMMS_TIMEOUT
Communication time out occurred.The test session remains open.

Failed to launch test node TES_xx
EXTIF_OS.START_APPLICATION:Operation OS_BINDING.START_APPLICATION raised excepti
Failed to invoke CGS script.

–> Check that the process creation server is running,
–> start test system may solve the problem.The present state of test node TES_xx is unknown.

Failed to open file.

TSCV_TRDB_SELECTOR.OPCS_SHOW_SESSION:Exception raised: USE_ERROR
Opening file /gsaf_home/4.1.1/tscv/data/session_view_file.tmp for read or write failed.

Failed to initialize CGS connections.”,

EXTIF_TSS.VERIFY_TSS_INTERFACE:The TSS interface returned
TSP_IF_NOT_INITIALISED”,
The CGS services are required to run in order
to provide for CGS inter–application communication.
The TSS interface has not been initialized!
None of the CGS application can be controlled by TSCV.

Failed to log to DBS, not connected!

Event: TSCV is stopping.

Failed to setup test node TES_xx

EXTIF_TES.INIT_TES:Operation TES_RPI.INIT returned COMMUNICATION_ERROR”,
Communication problem with the test node.
Try to shut–down and then restart the test node.
The test node TES_xx will not be setup, but will participate in this test configuration.

Failed to shut–down node(s)!

EXTIF_OS.SHUTDOWN_NODE:POSIX_ERROR
Failed to finally shutdown the CGS services on
the nodes: cgs–test1 csf_hp2,
using script stop_cgs_node located at $GSAF_HOME/cgs/bin/common.
Failed to invoke CGS script using remote shell. Verify access rights and correct the environment.

Failed to shut–down the Test Result DB.

EXTIF_DBS.STOP_CENTRAL_DBS:Operation DBS_RPI.STOP_CENTRAL returned
DBS_COMMUNICATION_ERROR
The error indicates that the TRDB has to be started.
Invoking check status for the test configuration may solve the problem.

Failed to start–up nodes

EXTIF_OS.STARTUP_NODE:POSIX_ERROR”,
Failed to start the CGS services using script start_cgs_node located at
$GSAF_HOME/cgs/bin/common.
Unless the CGS services are not running correctly, the test node cannot be operated.

Failed to unlock TSCV.

TSCV_LOCK.UNLOCK:Exception raised: NAME_ERROR
The TSCV lock file is probably not deleted
An internal TSCV program error was detected. Please report this error to the system manager.
This will not cause any problems. Even with a lock file present, TSCV will start if it is detected that the lock file is unchanged

Failed to update the TSCV lock file.

T_ALIVE_TASK:NAME_ERROR
The TSCV lock file ensures to start only one TSCV instance at a time. Cyclically updating the lock file located at $GSAF_HOME/tscv/data failed.
Using more that one TSCV instance at a time may result in conflicting test system setups.

Failed to verify status of test node TES_xx!
EXTIF_TES.GET_TES_STATUS:Operation TES_RPI.READ_STATUS
returned NO_SERVER_CONNEC
TES_xx is not running any more.
The status will be rendered ’No contact’

Loading test configuration.
In module: TC_SELECTOR.LOAD_TC_TEMPLATE”,
The test configuration contains node(s) that is not defined in the System Topology Table.
The missing node(s): HCI_03

No CCU is set in the Mission Data Base.
The status of the active test configuration no. <n> was verified.
Further set–up of TES instances will not be possible for this
test configuration unless stopped and set–up again.

Problem with connection to DBS: CONNECT_PROBLEM Stop and restart TSCV
Unexpected error occured. in module EXTIF_MDB.CLOSE_CONNECTION Symptom:
USE_ERROR. Please issue SPR!
—> In TSCV engineering version: Wrong sequence in DBS startup
—> Restart TSCV

Setup with node not in System Topology.”,
In module: L_TCS_CONTROLLER.OPCS_SETUP”,
Node HCI_xx is participating in the test configuration
to be setup, it is not, however, found in the System Topology table.

SMT error (1300) on host gsrf_dbs. Already member of a SMT domain
—> Maybe the TSP process on the node was already initialised in a
—> previous setup —> ignore as long as the previous setup was the same

SMT error (1302) on host gsrf_dbs. SMT–client(s) not reachable
—> Maybe the TSP process on the node is not runningwas already initialised in a
—> previous setup —> ignore as long as the previous setup was the same

SMT service is not setup on <node>!”,
In module: MAIN.CHECK_TSP_STATUS”,
All SMT based operation will not function on this node.
Checking status of the test configuration may solve the problem.
You may either continue checking the TSP status on remaining nodes, continue but suppress warnings or cancel further TSP checking.

Status of TRDB has changed.
The test session MARTIN associated to test configuration no. <n> seems to be closed since last time TSCV ran.
All related events will be logged to the default test session
You may create a new test session for this configuration

The test configuration is active!
In module: L_TCS_CONTROLLER.OPCS_SHUTDOWN_TES”,
Nodes hosting the DBS, the local TSCV or in use by
other active test configurations are excluded.”);

Status check failed.
EXTIF_DBS.GET_DBS_STATUS:Operation DBS_RPI.GET_TRDB_INFO

returned COMMS_TIME_OUT",TRDB availability was to be checked.
A problem was encountered when contacting DBS. Communication time out occurred.
Check network and communication services.

Test configuration(s) are active!
In module: L_TCS_CONTROLLER.OPCS_SHUTDOWN_TEST_SYSTEM
Confirm shut–down of the test system.

Test configuration has been modified!
All modifications will be lost when unloading the test configuration.

Test configuration no <n> is undefined
– REPOSITORY_TC.RETRIEVE_OLD_NODE_INIT_SYNCH_STATUS_:NAME_ERROR",
The description of the idle test configuration is not available in the Mission DB.
Program exception was raised when accessing save files for reading data.
The test configuration can not be used for further tests,but can still be viewed.
Please unload it

– L_TCS_CONTROLLER.OPCS_SETUP:Setup Denied",
The description of the test configuration is not available in the Mission DB.
An internal TSCV program error was detected.
Please report this error to the system manager. Setup can not be performed

Test node TES_xx rejected the stop request!
In module: L_TCS_TES_COMMANDING.OPCS_STOP_TEST

Test session <name> is already used.
L_TCS_SESSION_MANAGEMENT_SELECTOR.OPCS_APPLY:Session <name> already used.
No test session is created.

The test configuration is active!
In module: L_TCS_CONTROLLER.OPCS_SHUTDOWN_TES
Nodes hosting the DBS, the local TSCV or in use by other active test configurations are excluded.

Unable to read status for TES_01.

Unexpected status of TES_xx!
The test nodes were interrogated for current status.
Node TES_xx indicates that the status has changed from when TSCV was last shut–down.

Unexpected program error (check icon file).
in module GUI_CONTROLLER.CREATE_TSCV_ICON Symptom: WIN_ERROR. Please issue SPR!

—> installation problem (icon file not found)
—> can be ignored during integration of CGS

D-3.2.2.4 TCSV error messages in Pop Up Window

Internal program error
This message can appear, if operating system resources like disk space, swap space, etc. are not available. Call the system operator to check it.

Figure 15 : *The Internal Error window*

D-3.3 Test Execution

D-3.3.1 Messages from HCI (Workstation)

D-3.3.1.1 Messages on Console Window

**** MAIN PROGRAM ABANDONED — EXCEPTION ”constraint_error” RAISED**

—> if the workstation your are running your X–Server on is in a
—> bad state, this exception could occur
—> Re–Initialise your openlook env. or re–boot your workstation
—> Are the resources required exceeding the resource available ?
(e.g. is Text.MaxDocumentSize: 400000 set ?)

**** MAIN PROGRAM ABANDONED — EXCEPTION ”CONF_CODE_NOT_FOUND” RAISED**

—> The TSS configuration file is not present or contains
—> wrong data
—> Check in $TSS_HOME/config/tss_conf_file
—> should contain (V4.1):
NTP_STATUS_FILE /usr/tmp/ntp_status
SHARED_MEM_KEY_FILE /config/tss_shmkeyfile
SEMAPHORE_KEY_FILE /config/tss_semkeyfile
SMT_MONITOR_DELAY 0.1

D-3.3.1.2 Messages in Message Window

Alarm was not acknowledged!
Data Base Server Node: Alarm was not acknowledged!
—> In the system advisory window the status of the DB Server is
—> not ok. In the window the user did not acknowledge (i.e.
—> check box was not activated).
—> Ignore or activate

Alarm was not acknowledged!
<Sunsystem Name of Node>: Alarm was not acknowledged!
—> In the system advisory window the status of the Test Node where the subsystem is defined for
—> is not ok (i.e. there was at least one value out of limit)
—> In the window the user did not acknowledge (i.e.
—> check box was not activated).
—> Ignore or activate

Alarm acknowledged!
Data Base Server Node: Alarm acknowledged!
—> The operator has acknowledged the NOT_OK status in the system
—> advisory by activating the checkbox

Can't load synoptic display!",
Synoptic display:

CCU consistency not guaranteed.
The loaded CCU is not in a valid status. Please use the Consistency Checker for the CCU.
—> When connecting to the CCU, HCI gets the status reported from
—> the MDB. Means, that the consistency checker has not yet been run.
—> Can be ignored, if the status of the data is known. If not, the consistency
—> checker should be run.

CCU consistency status is local invalid
—> When connecting to the CCU, HCI gets the status reported from
—> the MDB. Means, that the consistency checker has found errors.
—> Can be ignored, if the status of the data is known and the errors reported
—> by the consistency checker can be ignored.

Checkout of "DVtools" failed
DVtools: cannot find license file (No such file or directory)
Product "DV–Tools" not validated — Error code 909
—> There is no license for your environment for the DATAVIEWS tool
—> Check with your system administration
—> Check if the License Server of DataViews is running

Command Facility: Illegal default path!
Ignored illegal default path <path>!
—> problem during screen setup when loading the command window:
—> the given default path was not a valid path in the selected CCU
—> Path must be virtual,CDU,system tree node
—> (i.e. must have attribute 'path_select'
—> B2 problem (17.1.96): CDU is not accepted in screen setup

Comms error in DBS_COMMS.INIT_LOCAL

SETUP_COMMUNICATION_CONFIGURATION returns NW_BADBINDPORT
—> when starting up HCI: could mean that another HCI is
—> already active on the same node !

Connected to node (TES_xx).,

Connected to <pathname> (TES_xx).”);

Connecting to configuration control u...”,

Element configuration is <element>, mission is <MISSION>, system tree version is <version>, CCU version is <pathname> <CCU Name> <Version>”);

Could not connect to test node \EUREC...

Could not connect to test node \EURECA\EGSE\PAULS_TEST\TEST_NODE (TES_01)!
—> the indicated test node was in the test configuration
—> but was not in the correct mode when HCI tried to connect to it

Could not load message definitions!

Error status was DEFINITIONS_NOT_FOUND!
—> During initialisation HCI loads data from file
—> $CGS_HOME/config/errmsg_defs.dat
—> This file has not been found

Could not load test configuration!

–Internal error!
—> HCI did not find a correct Test Configuration in file
—> $TSCV_HOME/data/cgs_test_configuration<n>.dat

–This Online Test Control (HCI) instance is not participating!

Could not request data!

No test node available for that data!
—> request for an enditem has been done
—> The enditem is not provided/monitored by any test node
—> Is the enditem loaded to a test node ?
—> Check Test Configuration and Generated Load_Scoe files
—> Is HCI connnected tothe test node which provides the data ?
—> was there an error during the connection ?

Could not start Online Test Control!

This Online Test Control (HCI_xx) is not participating in an executing test configuration!
—> There is no test configuration active, where the HCI_xx is participating
—> Start Test Configuration via TSCV first !

Disconnected from node (TES_xx).”,

Disconnected from test node ...\\TEST_NODE_02 (TES_02).”);

Exited Online Test Control.

Online Test Control started

Started by user <user>
—> HCI has been started successfully

D-3.3.2 Messages from TES (Test Node)

On Standard Output Device (e.g. console) where TES was started:

RUNTIME SYSTEM ERROR IN EXCEPTION HANDLING: Attempting traceback:
Program terminated by an exception propagated out of the main subprogram.
Exception raised : **CONSTRAINT_ERROR**
The exception was due to the following error: **RANGE_ERROR**
——— Stack trace of the propagation in the main stack —————
Compilation unit name Scope name Scope kind Line number
ALSYS_ADA_RUNTIME.EXCEPTI RAISE_FROM_RUNTIME SUBPROGRAM 2861
ON_MANAGER.RAISE_FROM_RUNTIME
(the above from original line 2511 of inlined or instantiated unit)
ALSYS_ADA_RUNTIME.EXCEPTI RAISE_EXCEPTION SUBPROGRAM 2632
ON_MANAGER
SYSTEM_ERROR: **Error during unwind of stack frames. Program aborted**.
/GSAF_HOME/tes/bin/common/start_tes: 15566 Killed

—> The elaboration of the program failed: Possible Reason:
—> Syntax error in TES_CONFIG_FILE
—> Check for blank or invalid lines etc.; correct
—> restart TES
—> If still same error: Check other elaboration parts of TES

In the message window:

Note: In the following, the type and group of each message is specified. When sent to the Message Window, the Type is mapped to the Message Window's classification scheme as follows:

ALRM	–> FATAL
EXC	–> SEVERE
ERR	–> ORDINARY
any other	–> ADVISORY

TYPE GROUP Text

ALRM MON ”Danger high limit monitoring exception”
”Danger high limit monitoring exception at <TIME> Raw val: <VALUE>
Eng val: <VALUE> Measurement: <PATHNAME>”
ALRM MON ”Danger high limit monitoring exception”
”Danger high limit monitoring exception at <TIME> Eng val: <VALUE>
SW Variable: <PATHNAME>”
ALRM MON ”Danger high limit monitoring exception”
”Danger high limit monitoring exception at <TIME> Eng val: <VALUE>
Derived value: <PATHNAME>”
ALRM MON ”Danger low limit monitoring exception”
”Danger low limit monitoring exception at <TIME> Raw val: <VALUE>
Eng val: <VALUE> Measurement: <PATHNAME>”
ALRM MON ”Danger low limit monitoring exception”
”Danger high limit monitoring exception at <TIME> Eng val: <VALUE>

SW Variable: <PATHNAME>”
ALRM MON ”Danger low limit monitoring exception”
”Danger high limit monitoring exception at <TIME> Eng val: <VALUE>
Derived value: <PATHNAME>”
ALRM MON ”Danger delta limit monitoring exception”
”Danger delta limit monitoring exception at <TIME> new / old Eng val:
VALUE> / <VALUE> <Measurement/SW Variable/Derived value>: <PATHNAME>”

ALRM TES ”<SAS_NAME> disconnected by TES”
”The application <SAS_NAME> got automatically disconnected, because it
did not check for commands sent to it within time. The disconnect timeout
can be configured by altering parameter API_CONTROLLER.
TIMEOUT_PERIOD_FOR_READ_CMD in the TES_CONFIG_FILE.”
ALRM TES ”Error creating the local SID table”,
”Not enough memory could be found to create the local SID table for data distribution”
ALRM TES ”Error creating the remote SID table”,
”The size of the remote SID table is not large enough. Please stop the test system, increase the
parameter DISTRIBUTION_TABLE.MAX_NB_SID in the TES_CONFIG_FILE and restart
the system”

TYPE GROUP Text

EXC MON ”<CODED_ERROR_TEXT1>”
”<CODED_ERROR_TEXT2>”
EXC MON ”Nominal high limit monitoring exception”
”Nominal high limit monitoring exception at <TIME> Raw val: <VALUE>
Eng val: <VALUE> Measurement: <PATHNAME>”
EXC MON ”Nominal high limit monitoring exception”
”Nominal high limit monitoring exception at <TIME> Eng val: <VALUE>
SW Variable: <PATHNAME>”
EXC MON ”Nominal high limit monitoring exception”
”Nominal high limit monitoring exception at <TIME> Eng val: <VALUE>
Derived value: <PATHNAME>”
EXC MON ”Nominal low limit monitoring exception”
”Nominal low limit monitoring exception at <TIME> Raw val: <VALUE>
Eng val: <VALUE> Measurement: <PATHNAME>”
EXC MON ”Nominal low limit monitoring exception”
”Nominal low limit monitoring exception at <TIME> Eng val: <VALUE>
SW Variable: <PATHNAME>”
EXC MON ”Nominal low limit monitoring exception”
”Nominal low limit monitoring exception at <TIME> Eng val: <VALUE>
Derived value: <PATHNAME>”
EXC MON ”Nominal delta limit monitoring exception”
”Nominal delta limit monitoring exception at <TIME> new / old Eng val:
VALUE> / <VALUE> <Measurement/SW Variable/Derived value>: <PATHNAME>”
EXC MON ”Expected value monitoring exception”
”Expected value monitoring exception at <TIME> Raw val: <VALUE>
Eng val: <VALUE> Measurement: <PATHNAME>”
EXC MON ”Expected value monitoring exception”
”Expected value monitoring exception at <TIME> Eng val: <VALUE>

SW Variable: <PATHNAME>”
EXC MON ”Expected value monitoring exception”
”Expected value monitoring exception at <TIME> Eng val: <VALUE>
Derived value: <PATHNAME>”

TYPE GROUP Text

ERR ARCH ”Automatic enable archiving.”
”Automatic enable archiving after system error. Archive queue is broken. Complete evaluation or replay of archive session is impossible.”
ERR ARCH ”Could not store archive file.”
”Failure calling DBS.Store_Raw_Data_File.”
ERR ARCH ”Could not create archive file.”
”Could not create archive file. Archiving disabled.”
ERR ARCH ”Could not close archive file.”
”The archive file <FILENAME> could not be closed.”
ERR ARCH ”Could not close archive file.”
”The archive file <FILENAME> could not be closed. Archiving disabled.”
ERR ARCH ”Could not create archive file.”
”Unexpected error while creating automatically a new archive file. Archiving disabled.”
ERR ARCH ”Illegal archive cycle.”
””
ERR ARCH ”Could not archive ADU.”
”Could not archive ADU. Archiving is disabled.”
ERR ARCH ”Could not archive ADU–request.”
”Could not archive ADU–request. Archiving is disabled.”
ERR ARCH ”Could not archive GDU.”
”Could not archive GDU. Archiving is disabled.”
ERR ARCH ”Could not archive SMT update.”
”Could not archive SMT update. Archiving is disabled.”

WRN ARCH ”Warning: Disk–space below limit.”
”available disk–space is: <NNNN> bytes.”
WRN ARCH ”Warning: Could not read free disk space.”
”TES internal error : Failure reading free disk space from System_Server: <NNNN>”

TYPE GROUP Text

ERR CDB ”Error in Loading Database”
”There was an error when unpacking measurement data from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking gdu data from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking adu data from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking monitor lists from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking gdu lists from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking simulated values file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking Automated Procedures from File”

ERR CDB ”Error in Loading Database”
”There was an error when unpacking node defs from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking sas names from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking user msg from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking user libraries from file”
ERR CDB ”Error in Loading Database”
”$MDA_HOME/data is not a directory”
ERR CDB ”Error in Loading Database”
”$MDA_HOME/data does not exist”
ERR CDB ”Error in Loading Database”
”A directory could not be created in the MDA data directory”
ERR CDB ”Error in Loading Database”
”A directory was missing in the MDA data directory”
ERR CDB ”Error in Loading Database”
”A file from the MDA data directory could not be opened”
ERR CDB ”Error in Loading Database”
”Error when closing file”
ERR CDB ”Error in Loading Database”
”Error when reading file”
ERR CDB ”Error in Loading Database”
”No node name in load point string”
ERR CDB ”Error in Loading Database”
”Load point pathname is missing”
ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because alternative is UNDEFINED”
ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because acquisition alternative is UNDEFINED”
ERR CDB ”Error in configuration data.”
”<PATHNAME> has an invalid calibration definition and therefore will never get a valid value.”
ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because the engineering value is of an unknown type.”
ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because the raw value is of an unknown type.”
ERR CDB ”Error in configuration data.”
”<PATHNAME> has an inconsistent default value type.”
ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because the HK value with ID: <NNNN>
is not supported. Please check in the MDB”
ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because the HK value with ID: <NNNN>
has already been linked by <PATHNAME>”
ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because there is a type mismatch between
the HK value with ID: <NNNN> and the software variable.”
ERR CDB ”Error in configuration data.”

”<PATHNAME> is discarded because the enditem type is not a measurement, software variable or derived value”
ERR CDB ”Error in configuration data.”
”Monitoring will not be performed on <PATHNAME> because no limit set is defined”
ERR CDB ”Error in configuration data.”
”Monitoring will not be performed on <PATHNAME> because limit set <NNNN> is invalid.”
ERR CDB ”Error in configuration data.”
”The enditem <PATHNAME> in the monitor list <PATHNAME> is not known.”
ERR CDB ”Error in configuration data.”
”<PATHNAME> will not be processed from ADU <PATHNAME> because it is not defined in the test node.”
ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because it is not acquired by an ADU.”
ERR CDB ”erroneous derived value”
”derived value <PATHNAME> is invalid because it has a cyclic dependency with <PATHNAME>”
ERR CDB ”erroneous derived value”
”derived value <PATHNAME> is erroneous because it uses <PATHNAME> that is erroneous”
ERR CDB ”erroneous derived value”
”derived value <PATHNAME> is erroneous because it uses <PATHNAME> that is not existing in the test node”
ERR CDB ”erroneous derived value”
”derived value <PATHNAME> is erroneous. An error occurred during initialisation”
ERR CDB ”error in dependency table”
”derived value <PATHNAME> linked to the software variable <PATHNAME> is not existing”
ERR CDB ”error in dependency table”
”<PATHNAME> is identified as a software variable in the dependency table but it has not been loaded”
ERR CDB ”error in dependency table”
”derived value <PATHNAME> linked to the adu <PATHNAME> is not existing”
ERR CDB ”error in dependency table”
”<PATHNAME> is identified as an ADU in the dependency table but it has not been loaded”
ERR CDB ”Error in configuration data.”
”The TC Verification definition of <PATHNAME> is incorrect and will be discarded : item at position <NNNN> is unknown on the test node.”
ERR CDB ”Error in configuration data.”
”The enditem <PATHNAME> in the GDU list <PATHNAME> is not known.”
ERR CDB ”Error in Loading SW command Database”
”There was an error when unpacking swop command data from file”
ERR CDB ”Error in Loading SW command Database”
”There was an error when unpacking application ID data from file”
ERR CDB ”Error in Loading SW command Database”
”There was an error when unpacking response packet data from file”
ERR CDB ”Error in Loading SW command Database”
”$MDA_HOME/data is not a directory”
ERR CDB ”Error in Loading SW command Database”
”$MDA_HOME/data does not exist”
ERR CDB ”Error in Loading SW command Database”

”A directory could not be created in the MDA data directory”
ERR CDB ”Error in Loading SW command Database”
”A directory was missing in the MDA data directory”
ERR CDB ”Error in Loading SW command Database”
”A file from the MDA data directory could not be opened”
ERR CDB ”Error in Loading SW command Database”
”Error when closing file”
ERR CDB ”Error in Loading SW command Database”
”Error when reading file”
ERR CDB ”Error in Loading SW command Database”
”No node name in load point string”
ERR CDB ”Error in Loading SW command Database”
”Load point pathname is missing”
ERR CDB ”Error in Loading Database”
”$MPS_HOME is no directory”
ERR CDB ”Error in Loading Database”
”$MPS_HOME not defined”
ERR CDB ”Error in Loading Database”
”MDB directory structure is incomplete”
ERR CDB ”Error in Loading Database”
”Directory could not be created on $MDB_HOME”
ERR CDB ”Error in Loading Database”
”Error within opening the Load SCOE file. Verify that Load SCOE files have been created”
ERR CDB ”Error in Loading Database”
”Error within closing the Load SCOE file”
ERR CDB ”Error in Loading Database”
”Read from Load SCOE file error”
ERR CDB ”Error in Loading Database”
”Error within Load SCOE file lock/unlock”
ERR CDB ”Error in Loading Database”
”No node name in load point string”
ERR CDB ”Error in Loading Database”
”Load point pathname is missing”
ERR CDB ”Error in Loading Database”
”Probably a memory problem.”

TYPE GROUP Text

ERR COND ”Could not start AP.”
”AP <PATHNAME> should have been started as a result of a condition,
but it cannot be loaded from the MDB”
ERR COND ”Could not start AP.”
”AP <PATHNAME> should have been started as a result of a condition,
but it has parameters”
ERR COND ”Could not start AP.”
”AP <PATHNAME> should have been started as a result of a condition,
but it is already running”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : IN_RANGE conditions cannot be defined
interactively.”

ERR COND ”Erroneous condition”
”<PATHNAME> is not a scalar type and cannot have a IN_RANGE condition check”
ERR COND ”Erroneous condition”
”<PATHNAME> references itself for selecting a limit set on condition”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : <PATHNAME> has no monitoring limits.”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : The limit set <NNNN> is undefined for enditem <PATHNAME>”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : The limit set number <NNNN> is illegal”
ERR COND ”Erroneous condition”
”<PATHNAME> references itself for enabling / disabling the processing on condition”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : <PATHNAME> is a SW variable linked to an HK data and its processing cannot be enabed / disabed on condition”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : <PATHNAME> is unknown on the testnode”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : ITEM is unknown on the testnode”
ERR COND ”Erroneous condition”
”CONDITION_ITEM is unknown on the testnode”

INFO COND ”Condition triggerred.”
”Enabling the processing of <PATHNAME>”
INFO COND ”Condition triggerred.”
”Disabling the processing of <PATHNAME>”
INFO COND ”Condition triggerred.”
”Setting limit set <NNNN> for <PATHNAME>”
INFO COND ”Condition triggerred.”
”Starting AP <PATHNAME>”
INFO COND ”Enabling conditions.”
”The conditions defined by <PATHNAME> are enabled”
INFO COND ”Disabling conditions.”
”The conditions defined by <PATHNAME> are disabled”

TYPE GROUP Text

ERR DACQ ”Could not start simulation on adu”
”The raw value (sid) <SID_NUMBER> (type: <RAW_VALUE_TYPE>) is not of type INT, U_INT or FLOAT_EGSE”
ERR DACQ ”Wrong ADU definition in MDB.”
”The global length of the ADU in MDB is wrong. Please check the size of <PATHNAME> and its simulated counterpart in MDB.”
ERR DACQ ”Could not set raw value in simulated adu”
”The raw value (sid) <SID_NUMBER> (type: <RAW_VALUE_TYPE>) is not of type INT, U_INT, FLOAT_EGSE or BYTE_STREAM”
ERR DACQ ”Could not construct ADU.”
”Could not construct ADU. ADU is undefined.”

ERR DACQ ”Could not construct ADU.”
”Could not construct ADU. ADU type mismatch.”
ERR DACQ ”Could not construct ADU.”
”Could not construct ADU. Illegal SID in ADU description.”
ERR DACQ ”Could not construct ADU.”
”Could not construct ADU. ADU value not defined.”
ERR DACQ ”Could not construct ADU.”
”Could not construct ADU. Index out of range.”
ERR DACQ ”Nothing to enable for acquisition”
”Pathname <PATHNAME>”
ERR DACQ ”Could not request ADU”
”Could not request ADU <PATHNAME> from <SAS_NAME>”
ERR DACQ ”Could not disable ADU request”
”Could not disable ADU <PATHNAME> from <SAS_NAME>”
ERR DACQ ”Parameter of ENABLE_MONITORING ignored”
”Pathname <PATHNAME> is already acquired. Parameter ADU is ignored”
ERR DACQ ”SUPPLY_ADU not allowed in current mode”
”TES must be started before SUPPLY_ADU is called”
ERR DACQ ”Update of software variable failed.”
”<PATHNAME> is not a software variable. Request from <NAME> refused.”
ERR DACQ ”Update of software variable failed.”
”<PATHNAME> is a housekeeping value. Request from <NAME> refused.”
ERR DACQ ”Write Software value failed.”
”Type mismatch. <PATHNAME> is of type <TYPE>, but the type <TYPE> was tried inserted. Request from <NAME> refused.”
ERR DACQ ”Unsupported SINGLE_BIT type”
”The raw value at index <NNNN> of ADU <PATHNAME> is SINGLE_BIT, which is not supported. Use U_INT.”
ERR DACQ ”Unknown raw value alternative”
”Raw value alternative <TYPE> not suitable for <structured ADU/unstructured ADU/CCSDS packet> <PATHNAME>.”
ERR DACQ ”Invalid length field in CCSDS packet”
”Length of CCSDS_PACKET is invalid. Please check the length field in the primary header for <PATHNAME> in the MDB.”
ERR DACQ ”Unknown ADU type”
”Procedure GET_RAW_VALUE”
ERR DACQ ”Error when processing ADU”
”ADU: <PATHNAME>; measurement no. <NNNN>: exception <EXCEPTION_NAME>”
ERR DACQ ”Internal Error”
”Internal error: Received empty ADU.”
ERR DACQ ”Internal Error”
”Set new CCSDS APID returns unexpected error for <PATHNAME>. ”
ERR DACQ ”Incorrect checksum in ADU”
”The ADU <PATHNAME> has an incorrect checksum.”
ERR DACQ ”Data interruption”
”The ADU <PATHNAME> has been interrupted. Processing ADU with sequence number <NNNN>”
ERR DACQ ”Data suspended”

”The ADU <PATHNAME> is suspended.”
ERR DACQ ”Attempt to read value failed”
”<PATHNAME> is not a measurement, sw variable or derived value”
ERR DACQ ”Attempt to read value failed”
”Value of <PATHNAME> is invalid (not acquired or incorrect due to previous error.)”
ERR DACQ ”Attempt to read raw value failed”
”<PATHNAME> is not a measurement.”
ERR DACQ ”Attempt to read raw value failed”
”The raw value of <PATHNAME> has not yet been acquired.”
ERR DACQ ”Failed to get monitoring status.”
”<PATHNAME> is not a measurement, sw variable or derived value”
ERR DACQ ”Stop acquisition rejected”
”<PATHNAME> is monitored and can therefore not be disabled for acquisition.”
ERR DACQ ”Failed to turn EVL <ON/OFF>”
”The engineering value logging cannot be changed. <PATHNAME> is unknown.”
ERR DACQ ”ADU service announced again”
”The application <SAS_NAME> has already announced its ADU service.”
ERR DACQ ”Withdraw ADU service failed.”
”Withdraw ADU service failed because the application <SAS_NAME> as not announced its ADU service.
ERR DACQ ”Cannot set SAS reference for ADU”
”The ADU <PATHNAME> is currently acquired. Its SAS reference cannot be changed”
ERR DACQ ”Cannot set SAS reference for ADU”
”Parameter OLD_SAS_NAME does not match the current SAS name for <PATHNAME>”
ERR DACQ ”Data conversion failure”
”String too long (pathname =<PATHNAME>)”
ERR DACQ ”Data conversion failure”
”String too long (hk_id =<NNNN>)”
ERR DACQ ”Internal Error”
”Internal error: ADU with ADU_ID <NNNN> not in local database”
ERR DACQ ”Could not read TRDB INFO HK–values”
”DBS TRDB INFO HK–values will not be written. Received return status <STATUS> from DBS_RPI_FOR_TEST_NODE.GET_TRDB_INFO.”
ERR DACQ ”Could not read disk HK–values”
”DBS disk HK–values will not be written. Received return status <STATUS> from DBS_RPI_FOR_TEST_NODE.GET_DISK_PARAMETERS.”
ERR DACQ ”Could not read Printer HK–values”
”DBS Printer HK–values will not be written. Received return status <STATUS> from DBS_RPI_FOR_TEST_NODE.CHECK_PRINT.”
ERR DACQ ”Could not read oracle HK–values”
”DBS oracle statistics HK–values will not be written. Received return status <STATUS> from DBS_RPI_FOR_TEST_NODE.GET_ORACLE_STATISTICS.”
ERR DACQ ”EGSE node not found”
”EGSE node with pathname <PATHNAME> not found in the internal TES database”
ERR DACQ ”EGSE sw not found in internal database”
”The EGSE sw: <NNNN>”
ERR DACQ ”Internal Error”
”Internal error: <PATHNAME> is not an ADU.”

ERR DACQ ”Internal Error”
”Internal error: ADU <PATHNAME> not in local database.”

ERR DACQ ”Read Value Error”
”The supplier <SAS_NAME> of the ADU <PATHNAME> has not announced its ADU service.”

ERR DACQ ”Could not set value in ADU.”
”Could not set value in ADU. Measurement undefined (SID: <NNNN>). Pathname: <PATHNAME>”

ERR DACQ ”Could not set value in ADU.”
”Could not set value in ADU. Simulator is stopped.”

ERR DACQ ”Could not set value in ADU.”
”Could not set value in ADU. Measurement unknown (SID: <NNNN>). Pathname: <PATHNAME>”

ERR DACQ ”Could not set value in ADU.”
”Could not set value in ADU. Illegal SID.”

ERR DACQ ”Could not set bits in ADU.”
”Could not set bits in ADU. Constraint error: Bit location out–of–range.”

ERR DACQ ”Could not set bits in ADU.”
”Could not set bits in ADU. Number of modifying bits must be greater than 0 and less or equal to <NNNN>.”

ERR DACQ ”Could not set bits in ADU.”
”Simulator is stopped.”

ERR DACQ ”Could not set bits in ADU.”
”Could not set bits in ADU. ADU not found.”

ERR DACQ ”Could not set bits in ADU.”
”Could not set bits in ADU. Illegal SID.”

ERR DACQ ”Could not start simulate ADU.”
”Could not start simulate ADU <PATHNAME>. Simulation is suspended.”

ERR DACQ ”Could not start simulate ADU.”
”Could not start simulate ADU <PATHNAME>. Simulator is stopped.”

ERR DACQ ”ADU not simulated.”
”ADU <PATHNAME> is not being simulated.”

ERR DACQ ”Could not stop simulation ADU.”
”Could not stop simulation ADU <PATHNAME>. Simulation is suspended.”

ERR DACQ ”Could not stop simulation ADU.”
”Could not stop simulation ADU <PATHNAME>. Simulator is stopped.”

ERR DACQ ”Could not send simulated ADU.”
”Could not send simulated ADU. ADU <PATHNAME> is not being simulated.”

ERR DACQ ”Could not send simulated ADU.”
”Could not send simulated ADU <PATHNAME>. Simulation is suspended.”

ERR DACQ ”Could not send simulated ADU.”
”Could not send simulated ADU <PATHNAME>. Simulator is stopped.”

ERR DACQ ”ADU request failed”
”The sending of an disable ADU request for the response packet <PATHNAME> to <SAS_NAME> failed with error <STRING>”

ERR DACQ ”Software command timeout”
”Response <PATHNAME> has not been received in time for software command <PATHNAME>”

ERR DACQ ”Incorrect checksum in response packet”
”The response packet <PATHNAME> has an incorrect checksum.”
ERR DACQ ”Wrong transaction_id in response packet”
”Response packet <PATHNAME> received with transaction id=<HHHH>,
expected=<HHHH> (the hexadecimal repr. of the transaction_id
extracted from the return packet resp. the CCSDS primary header
of the telecommand”
ERR DACQ ”Invalid Response Packet”
”Response packet <PATHNAME> contained an invalid CCSDS packet or
invalid return parameters”

WRN DACQ ”Data packet overflow – packet discarded”
”Discarding data packet <PATHNAME>. Starting with sequence number <NNNN>.
The subsequent data packet of that type will be treated as interrupted”
WRN DACQ ”Write Value Error”
”The application <NAME> supplied the ADU <PATHNAME>, but the acquisition
of this ADU is not enabled.”
WRN DACQ ”Internal Error”
”SUPPLY_ADU returned with unexpected return status <NNNN>”

INFO DACQ ”ADU Service Announced”
”The application <SAS_NAME> has announced its ADU Service.”
—>The application (SAS) was initialised and announced its ADU service. It is ready now
—> to accept request for acquisition (i.e. enditems associated with this SAS may
—> acquired via START_ACQUISITION command**)**

INFO DACQ ”ADU Service Withdrawn”
”The application <SAS_NAME> has withdrawn its ADU Service.”
—>The SAS has withdrawn its ADU service, i.e. it will not accept any request for
—> enditem acquisition via the START_ACQUISITION command anymore
—> Most SAS will withdraw this service when switched to the RESET state or when
—> re–initialised or when being unloaded.

INFO DACQ ”New SAS reference for ADU”
”SAS reference has been set to <SAS_NAME> for <PATHNAME>”
INFO DACQ ”ROUTE_TO_SAS ignored”
”UCL system library routine ROUTE_TO_SAS ignored in replay mode.”
INFO DACQ ”New SAS reference for ADU’s”
”SAS reference has been set to <SAS_NAME> for all ADU’s under <PATHNAME>
that are currently not acquired.”
INFO DACQ ”New SAS reference for ADU’s”
”SAS reference has been set to <SAS_NAME> for all ADU’s under <PATHNAME>
that are currently not acquired and where the SAS reference was <SAS_NAME>.”
MSG DACQ ”Set new CCSDS APID”
”For <PATHNAME> set CCSDS APID to <VALUE>.”
MSG DACQ ”Set new CCSDS APID”
”For <PATHNAME> set CCSDS APID from <VALUE> to <VALUE>.”
MSG DACQ ”Set new CCSDS APID”
”For enditems in <PATHNAME> set CCSDS APID to <VALUE>.”

MSG DACQ ”Set new CCSDS APID”
”For enditems in <PATHNAME> set CCSDS APID from <VALUE> to <VALUE>.”
MSG DACQ ”Set new CCSDS APID”
”For enditems under <PATHNAME> set CCSDS APID to <VALUE>.”
MSG DACQ ”Set new CCSDS APID”
”For enditems under <PATHNAME> set CCSDS APID from <VALUE> to <VALUE>.”

TYPE GROUP Text

ERR DDS ”Dispatch of ADU failed”
”<PATHNAME>: disabling data dispatch due to HCI connection problem”
ERR DDS ”Dispatch of ADU failed”
”<PATHNAME>; dispatch return status: <NNNN>”
=> <dispatch_status>: status when dispatching ADU to HCI
ERR DDS ”Invalid item in data delivery request.”
”<PATHNAME> is not a measurement, sw variable or derived value. The item will be ignored.”
ERR DDS ”Invalid item in data delivery request.”
”The data requested with SID <NNNN> (pathname <PATHNAME>) is unknown. The item will be ignored.”
ERR DDS ”Invalid item in data delivery request.”
”The data requested is unknown. The item will be ignored.”
ERR DDS ”Error in Request for dispatch of HK data”
”Hk value <NNNN> does not exist.”
ERR DDS ”Request for ADU dispatch contains error.”
”<PATHNAME> is not an ADU enditem.”
ERR DDS ”Request for ADU dispatch contains error.”
”The data requested is unknown.”
ERR DDS ”Enditem is of wrong type.”
”<PATHNAME> is not an ADU enditem.”
ERR DDS ”Error dispatching data to HCI”
”<STATUS> returned by HCI when delivering data for request ID = <NNNN> of <HCI_NAME>”
=> if <status>: PROTOCOL_ERROR: one possible reason:
—> HCI was shutdown non–nominally when dispatch was
—> running
—> restart HCI
—> re–initialise TES (if errors are still displayed after
—> HCI restart)
ERR DDS ”Invalid request ID”
”Attempted to reuse an existing request ID (<NNNN>)”
ERR DDS ”Invalid data delivery request”
”Invalid data delivery request from <HCI_NAME>: Delivery type undefined.”
ERR DDS ”Repeated confirm”
”Request <NNNN> already confirmed”
ERR DDS ”DDS Stopped”
”Attempted to confirm delivery in idle mode”
ERR DDS ”Unknown request ID”
”Request ID <NNNN> is not a registered request”
ERR DDS ”Interruption of data dispatch to HCI”

”An overflow occurred when delivering DDS packet to <HCI_NAME>. A data interruption will occur.”

WRN DDS ”Internal consistency error”
”Missing request for <ADU/enditem/hk–item> <NNNN>”
WRN DDS ”Internal consistency error”
”Missing request list for <ADU/enditem/hk–item> <NNNN>”
WRN DDS ”Internal consistency error”
”Request ID <NNNN> not registered”

TYPE GROUP Text

ERR DGEN ”Communication error on Issue TC/stimuli”
”Issueing of <PATHNAME> failed du to a communication error”
ERR DGEN ”Communication error on Issue TC/stimuli”
”Issueing of <PATHNAME> failed du to a timeout situation. The timeout value (currently <NNNN> ms) is an input value to the issue command”
ERR DGEN ”Could not make GDU.”
”Enditem has not been authorized. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU.”
”Enditem is disabled. Pathname: <PATHNAME>”
ERR DGEN ”Issuing of GDU failed, SAS not connected”
”SAS not connected for GDU. Pathname: <PATHNAME>”
ERR DGEN ”Issuing of GDU failed, NACK received”
”SAS has send a NACK for GDU. Pathname: <PATHNAME>”
ERR DGEN ”Issuing of GDU failed”
”Issuing of GDU failed due to unexpected failure. Pathname: <PATHNAME>”
ERR DGEN ”Error in sending a message to an AP.”
”Application <SAS_NAME> tried to send a message to AP <PATHNAME>, but this pathname could not be translated to an SID. ”
ERR DGEN ”Error in request for raw value.”
”Application <SAS_NAME> tried to read the raw value of <PATHNAME>, but this pathname could not be translated to an SID. ”
ERR DGEN ”Error in request for engineering value.”
”Application <SAS_NAME> tried to read the eng. value of <PATHNAME>, but this pathname could not be translated to an SID. ”
ERR DGEN ”Error in request for engineering value.”
”Application <SAS_NAME> tried to write the eng. value of <PATHNAME>, but this pathname could not be translated to an SID. ”
ERR DGEN ”Initialisation Error”
”Error during initialisation of TES local datapool”
ERR DGEN ”Could not enable issueing”
”Simulation is suspended. Could not enable issueing.”
ERR DGEN ”Cannot enable GDU”
”Could not enable enditem <PATHNAME>”
ERR DGEN ”Could not disable issueing”
”Simulation is suspended. Could not disable issueing.”
ERR DGEN ”Cannot disable GDU”
”Could not disable enditem <PATHNAME>”
ERR DGEN ”Could not make GDU list”
”Incorrect parameter type definition. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU list”
”Wrong number of parameters to a gdu list. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU list”
”Incorrect parameter type definition. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU list”
”Index parameters for GDU list out of allowed ranges. Pathname: <PATHNAME>”
ERR DGEN ”Could not announce GDU service”
”Simulation is suspended. Could not announce the GDU service.”

ERR DGEN ”Could not withdraw GDU service”
”Simulation is suspended. Could not withdraw the GDU service.”
ERR DGEN ”Could not issue GDU”
”Simulation is suspended. Could not issue GDU : <PATHNAME>”
ERR DGEN ”Could not make GDU”
”GDU service not announced. Pathname: <PATHNAME>”
ERR DGEN ”Error during issue of GDU list”
”GDU list <PATHNAME>. Errors detailled in earlier messages”
ERR DGEN ”Could not make GDU.”
”The enditem is unknown. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU.”
”The enditem is not a local GDU. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Wrong number of parameters. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Wrong parameter type <TYPE> specified for GDU at position <NNNN>
Parameter type <UNSIGNED INTEGER/INTEGER/REAL/TIME/STATE CODE/
PATHNAME> expected. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Wrong parameter type specified for GDU at position <NNNN>
Parameter type <UNSIGNED INTEGER/INTEGER/REAL/STRING/TIME/
STATE CODE/PATHNAME> expected. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Wrong raw value type specified in MDB. Engineering values of types integer
can only be decalibrated to integer and unsigned integer raw values.
Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Wrong raw value type specified in MDB. Engineering values of types real
can only be decalibrated to float, integer and unsigned integer raw values.
Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Size of GDU parameter at position <NNNN> does not allow to insert the real
parameter. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Size of GDU parameter at position <NNNN> does not allow to insert the actual
string value. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Time parameter not defined (date part not set). Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Time parameter has to be greater than the TAI epoch specified in the
TES configuration file Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Size of GDU parameter at position <NNNN> does not allow to insert a time
value (40 bits needed). Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Size of GDU parameter at position <NNNN> does not allow to insert a statecode
value (64 bits needed). Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”

”Unsupported parameter type <TYPE> specified for parameter number : <NNNN>
Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Unsupported parameter type specified for parameter number : <NNNN>
Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Length error for parameter number <NNNN>. Value exceeds length of parameter or parameter exceeds the CCSDS packet length. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Length error for parameter number <NNNN>. Value exceeds length of parameter or parameter exceeds the binary packet length. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Onboard execution time contains a time but no date. Date required.
Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Onboard execution time has to be greater than the TAI epoch specified in the TES configuration file Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”GDU service not announced Pathname: <PATHNAME>”
— The addressed SAS did not anounce GDU service yet
ERR DGEN ”Could not make GDU”
”Incorrect default value. Real type expected. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Analog stimuli requires float parameters Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Incorrect default value. State code type expected. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”The parameter is not a discrete value Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Enditem is disabled Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”The decalibration parameter was out of range. Please check allowed range and types for this TC/stimuli in MDB. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”The decalibration definition is invalid. Please check the decalibration definition in MDB. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Invalid calibration type defined in MDB. Please check raw and engineering types for this TC/stimuli in MDB. Pathname: <PATHNAME>”
—> A decalibration was defined in the DB for the GDU.
—> The type defined there does not
—> match the type of the parameter given for the GDU
ERR DGEN ”Could not make GDU”
”Missing default value to TC/stimuli. Please provide a value as parameter or assign a default value in MDB. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Exception <EXCEPTION_NAME> was raised when construction a GDU.
Pathname: <PATHNAME>”

ERR DGEN ”Cannot set SAS name for GDU”
”Parameter OLD_SAS_NAME does not match the current SAS name for <PATHNAME>”
ERR DGEN ”Negative TAI epoch time”
”Current time has to be greater than the TAI epoch specified in the TES configuration file”
ERR DGEN ”Negative execution time”
”Onboard execution time has to be greater than the TAI epoch specified in the TES configuration file”
ERR DGEN ”Current mode wrong.”
”Set CCSDS end point not allowed in the current mode.”
ERR DGEN ”Invalid node name”
”The given pathname is not a CCSDS end point.”
ERR DGEN ”Current mode wrong.”
”Enable of SW command not allowed in the current mode.”
ERR DGEN ”Current mode wrong.”
”Disable of SW command not allowed in the current mode.”
ERR DGEN ”SW command for FLAP not found.”
”SW command <PATHNAME> for FLAP <PATHNAME> not found”
ERR DGEN ”SW command not found.”
”SW command <PATHNAME> not found”
ERR DGEN ”SW command disabled.”
”SW command <PATHNAME> disabled.”
ERR DGEN ”GDU service not announced.”
”The application <SAS_NAME> has not annonunced its GDU service.”
ERR DGEN ”ADU service not announced.”
”The application <SAS_NAME> has not annonunced its ADU service.”
ERR DGEN ”Ground Node not specified.”
”The Ground Node parameters was not given and the default Ground Node is not set.”
ERR DGEN ”Invalid node parameter.”
”Application id for the node pair (ground node: <PATHNAME> to onboard node: <PATHNAME>) could not be found.”
ERR DGEN ”Invalid node parameters.”
”Application id’s for the node pair (ground node: <PATHNAME> to onboard node: <PATHNAME>) could not be found.”
ERR DGEN ”Sending of SW command failed.”
”Sending of the SW command <PATHNAME> failed.”
ERR DGEN ”Set Device address failed”
”No APID is defined for end point pair : source <PATHNAME> / destination <PATHNAME>”
ERR DGEN ”Cannot set SAS name for SWOP”
”Parameter OLD_SAS_NAME does not match the current SAS name for <PATHNAME>”
ERR DGEN ”Cannot set SAS name for Response Packet”
”Parameter OLD_SAS_NAME does not match the current SAS name for <PATHNAME>”

WRN DGEN ”Warning : incomplete CCSDS packet”
”Value missing for time parameter. Parameter will not be set in the GDU. Pathname : <PATHNAME>”
WRN DGEN ”Warning : incomplete GDU – CCSDS packet”
”CCSDS packet without secondary header. Parameter ONBOARD_EXECUTION_TIME will not be set in the GDU. Pathname : <PATHNAME>”

WRN DGEN ”Parameter list truncated for SW command”
”The parameter part of the software command <PATHNAME> only contains <NNNN> Bytes. Not all IN parameters have been inserted.”
WRN DGEN ”Response packet too short”
”The OUT parameter part of the response packet for <PATHNAME> only contains <NNNN> Bytes. Remaining OUT parameters are set to default value.”

INFO DGEN ”ENABLE_ENDITEM ignored”
”UCL system library routine ENABLE_ENDITEM ignored in replay mode.”
INFO DGEN ”Enabled enditem”
”Enabled issuing enditem with pathname: <PATHNAME>”
INFO DGEN ”Enabled enditems”
”Enabled issuing of <NNNN> enditems contained in : <PATHNAME>”
INFO DGEN ”DISABLE_ENDITEM ignored”
”UCL system library routine DISABLE_ENDITEM ignored in replay mode.”
INFO DGEN ”Disabled enditem”
”Disabled issuing enditem with pathname: <PATHNAME>”
INFO DGEN ”Disabled enditems”
”Disabled issuing of <NNNN> enditems contained in : <PATHNAME>”
INFO DGEN ”GDU Service Announced”
”The application <SAS_NAME> has announced its GDU Service.”
INFO DGEN ”GDU Service Withdrawn”
”The application <SAS_NAME> has withdrawn its GDU Service.”
INFO DGEN ”ISSUE ignored”
”UCL system library routine ISSUE ignored in replay mode.”
INFO DGEN ”New SAS reference for GDU”
”SAS reference has been set to <SAS_NAME> for <PATHNAME>”
INFO DGEN ”New SAS reference for GDU’s”
”SAS reference has been set to <SAS_NAME> for all GDU’s referenced in <PATHNAME>”
INFO DGEN ”New SAS reference for GDU’s”
”SAS reference has been set to <SAS_NAME> for all GDU’s referenced in <PATHNAME> where the SAS reference was <SAS_NAME>”
INFO DGEN ”New SAS reference for GDU’s”
”SAS reference has been set to <SAS_NAME> for all GDU’s under <PATHNAME>”
INFO DGEN ”New SAS reference for GDU’s”
”SAS reference has been set to <SAS_NAME> for all GDU’s under <PATHNAME> where the SAS reference was <SAS_NAME>”
INFO DGEN ”Send GDU in sim mode”
”<PATHNAME>”
INFO DGEN ”Single GDU issued”
”GDU <PATHNAME> issued”
INFO DGEN ”List of GDU packets issued”
”Packet list <PATHNAME> issued”
INFO DGEN ”SW command sent.”
”SW command <PATHNAME> sent, transaction id=<HHHH> where <HHHH> is the hexadecimal of the CCSDS primary header of the telecommand”
INFO DGEN ”Device address changed for APID <NNNN>”
”New device address is <STRING>”
INFO DGEN ”New SAS reference for SWOP”

”SAS reference has been set to <SAS_NAME> for <PATHNAME>”
INFO DGEN ”New SAS reference for Response Packet”
”SAS reference has been set to <SAS_NAME> for <PATHNAME>”
INFO DGEN ”New SAS reference for SWOP items”
”SAS reference has been set to <SAS_NAME> for all SWOP’s and response packets under <PATHNAME>”
INFO DGEN ”New SAS reference for SWOP items”
”SAS reference has been set to <SAS_NAME> for all SWOP’s and response packets under <PATHNAME> where the SAS reference was <SAS_NAME>”

TYPE GROUP Text

ERR DGVF ”TC Verification failed”
”The verification of <PATHNAME> failed : <NNNN> check(s) failed.”
ERR DGVF ”TC Verification check failed”
”Command <PATHNAME> : enditem <PATHNAME> is not acquired”
ERR DGVF ”TC Verification check failed”
”Command <PATHNAME> : check of enditem <PATHNAME> failed – value is invalid”
ERR DGVF ”TC Verification check failed”
”Command <PATHNAME> : check of enditem <PATHNAME> failed – last value: <VALUE>”
ERR DGVF ”Could not get GDU verification status”
”Simulation is suspended. Could not GDU verification status : <PATHNAME>”
ERR DGVF ”Could not get GDU verification status.”
”The enditem is unknown. Pathname: <PATHNAME>”
ERR DGVF ”Could not get GDU verification status.”
”The enditem is not a local GDU. Pathname: <PATHNAME>”

WRN DGVF ”TC Verification aborted”
”TC verification for <PATHNAME> is aborted due to the stop of the test node.”
WRN DGVF ”TC Verification aborted”
”Command <PATHNAME> has been issued while the verification part of its last generation has not been completed. Previous TC Verification is aborted”

INFO DGVF ”TC Verification successful”
”The verification of <PATHNAME> is successful.”
INFO DGVF ”GET_VERIFICATION_STATUS ignored”
”UCL system library routine GET_VERIFICATION_STATUS ignored in replay mode.”

TYPE GROUP Text

ERR HCI ”Shutdown Error.”
”The status <STATUS> was received when disconnecting from HCI.”

TYPE GROUP Text

MSG LOG ”<AP_TEXT>”
”<AP_TEXT>”

TYPE GROUP Text

ERR MON ”Calibration Error”
”Calibration of <TYPE> to <TYPE> not supported for measurement <PATHNAME>.”
ERR MON ”Calibration Error”
”Raw or calibrated value out of defined ranges for measurement <PATHNAME>.”

ERR MON ”Calibration Error”
”Calibration description not properly defined for measurement <PATHNAME>.”
=> exceptions occured in CALIBRATION package
ERR MON ”Calibration Error”
”Enditem corrupted: Could not calibrate for measurement <PATHNAME>.”
=> in CALIBRATE_RAW_VALUE if unknown exception occurs
ERR MON ”Limit not set in Enable_monitoring”
”Limit set no. <NNNN> is undefined for enditem <PATHNAME>.”
ERR MON ”Change limits failed.”
”<PATHNAME> is not a measurement, sw variable or derived value”
ERR MON ”Change limits failed.”
”<PATHNAME> is not known.”
ERR MON ”Change limits failed.”
”<PATHNAME> has no monitoring limits.”
ERR MON ”Set nominal limit failed.”
”limit value outside integer range for <PATHNAME>”
ERR MON ”Set nominal limit failed.”
”Can not set nominal limits for nonscalar value <PATHNAME>”
ERR MON ”Set nominal limit failed.”
”New limit value inconsistent with the current limit set of <PATHNAME>”
ERR MON ”Set nominal limit failed.”
”Can not set nominal limits for non integer value <PATHNAME>”
ERR MON ”Set expected state code failed”
”<PATHNAME> is not a state code type enditem.”
ERR MON ”Failed to set expected string.”
”<PATHNAME> is not a string type enditem.”
ERR MON ”Set limit set failed.”
”The limit set <NNNN> is undefined for enditem <PATHNAME>”
ERR MON ”Set exception count failed”
”Exception count for <PATHNAME> not set (provided value too large.)”
ERR MON ”Invalid limits set.”
”The limit set number <NNNN> is illegal”
ERR MON ”Invalid delta limit.”
”The delta limit can not be negative.”
ERR MON ”Invalid exception count limit.”
”The exception count must be greater than zero.”
ERR MON ”Emergency AP not found.”
”Emergency AP <PATHNAME> was not loaded initially. The AP will be loaded from MDB.”
ERR MON ”Could not start emergency AP.”
”AP <PATHNAME> should have been started as a result of a monitoring exception, but it cannot be loaded from the MDB”
ERR MON ”Could not start emergency AP.”
”AP <PATHNAME> should have been started as a result of a monitoring exception, but it has parameters”
ERR MON ”Could not start emergency AP.”
”AP <PATHNAME> should have been started as a result of a monitoring exception, but it is already running”

INFO MON ”Enabling monitoring.”
”Enabling monitoring with limit set <NNNN> for enditem(s) <PATHNAME>”
INFO MON ”Disabling monitoring.”
”Disabling monitoring for enditem(s) <PATHNAME>”
INFO MON ”New nominal limit definition.”
”Nominal high limit for <PATHNAME> has been set.”
INFO MON ”New nominal limit definition.”
”Nominal low limit for <PATHNAME> has been set.”
INFO MON ”New nominal limit definition.”
”Nominal delta limit for <PATHNAME> has been set.”
INFO MON ”New expected state code.”
”Expected state code for <PATHNAME> has changed to <STATCODE>.”
INFO MON ”New expected Value”
”Expected value for <PATHNAME> has changed to <STRING>.”
INFO MON ”Setting new limit set.”
”Setting limit set <NNNN> for enditem(s) <PATHNAME>”
INFO MON ”New Exception Count”
”Exception Count for <PATHNAME> has changed to <NNNN>.”

TYPE GROUP Text

ERR REPL ”Replay initialisation error.”
”Replay initialisation error. Replayer must be initialised in replay–mode.
Replaying stopped.”
ERR REPL ”Replay initialisation error.”
”Replay initialisation error. The replay end time is before the start time.
Replaying stopped.”
ERR REPL ”Replay initialisation error.”
”Replay initialisation error. The replay start and end time is equal.
Replaying stopped.”
ERR REPL ”Replay initialisation error.”
”Replay initialisation error. Can not read archive file: <FILENAME>
Replaying stopped.”
ERR REPL ”End of archive file(s) reached.”
”End of archive file(s) reached. Replaying stopped.”
ERR REPL ”Exception raised.”
”Exception DURATION_OVERFLOW raised due to too long wait period
between two archive file items. Replaying stopped.”
ERR REPL ”Replaying stopped.”
”Replaying stopped, because an unexpected exception was raised.”
ERR REPL ”Replay initialisation error.”
”Exception DURATION_OVERFLOW raised due to too long wait period for
the first item to be replayed.”
ERR REPL ”Wrong TES mode.”
”Wrong TES mode. Not allowed to restart TES in replay mode without reinitialising.”

WRN REPL ”TES is late in replay.”
”TES is more than 5 seconds late in replay Possible cause: too many
data or speed too high”
WRN REPL ”Nothing to replay”

"If MTP, cannot set local time and SMT value. If test with several test nodes, synchronisation will not be achieved. Use local time setup instead"
WRN REPL "replay with non running SMT"
"If test with several test nodes, synchronisation will not be achieved. Use local time setup instead"
WRN REPL "Nothing to replay"
"If MTP, cannot set local time and SMT value. If test with several test nodes, synchronisation will not be achieved"
WRN REPL "Nothing to replay"
""

INFO REPL "Replaying stopped by the user."
"Replaying stopped by the user. Replaying stopped."
INFO REPL "Replaying of data stopped."
"Replaying of data stopped, because the end of the last archive file was reached. Replaying stopped."
INFO REPL "Replaying of last item done"
"Waiting for replay end time"
INFO REPL "Nothing to replay"
""
INFO REPL "Initial delay : <HH> hrs <MI> min <SS> sec."
"The first data to replay has been found in the archive at local time <DD>.<MM>.<YYYY> <HH>:<MI>:<SS>. Consider a reduction of the time frame to replay if the initial delay appears to be too long."
INFO REPL "Initial delay : <MI> min <SS> sec."
"The first data to replay has been found in the archive at local time <DD>.<MM>.<YYYY> <HH>:<MI>:<SS>. Consider a reduction of the time frame to replay if the initial delay appears to be too long."
INFO REPL "Initial delay : <SS> sec."
"The first data to replay has been found in the archive at local time <DD>.<MM>.<YYYY> <HH>:<MI>:<SS>."
INFO REPL "Waiting for replay end time"
"Waiting for replay end time"
INFO REPL "Replayed <GDU_TYPE> command"
"Replayed command <PATHNAME>, orig. time was <TIME> [LT] & <TIME> [SMT]."
INFO REPL "Started acquisition of a data packet"
"Packet <PATHNAME>, orig. time was <TIME> [LT] & <TIME> [SMT]."
INFO REPL "Stopped acquisition of a data packet"
"Packet <PATHNAME>, orig. time was <TIME> [LT] & <TIME> [SMT]."
INFO REPL "Suspended replaying."
"Suspended replaying on <NODE>"
INFO REPL "Resumed replaying."
"Resumed replaying on <NODE>"

TYPE GROUP Text

ERR SAS "acknowledge code out of range",
"The acknowledge code delivered by SAS <NAME> is out of range."
ERR SAS "Link state control error"

”SAS name unknown”
ERR SAS ”Not connected”
”Application name: <SAS_NAME>”
ERR SAS ”Sending a command to application failed”
”The command <COMMAND> did not get acknowledged or was not read by <SAS_NAME> within the timeout period of: <NNNN> ms”
ERR SAS ”Error in communication with application”
”Sending <COMMAND> to application <SAS_NAME> returned with status: <STATUS>”
ERR SAS ”Multiple load”
”Application <SAS_NAME> already running”
ERR SAS ”OS_BINDING error”
”Error: Create_Process failed, file <FILENAME> not found”
=> SAS could not be started. Executable not found
ERR SAS ”OS_BINDING error”
”Error: Create_Process failed for file <FILENAME>. Host name not known on this node”
=> SAS could not be started. Host not found
ERR SAS ”OS_BINDING error”
”Error: Create_Process failed for file <FILENAME>. The server process was not found on the indicated host”
=> The process_creation_server is not running or not responding (time–out)
ERR SAS ”OS_BINDING error”
”Error: Create_Process failed for file <FILENAME>. The server process was killed”
=> The process_creation_server is not running or crashed
ERR SAS ”OS_BINDING error”
”Error: Create_Process failed for file <FILENAME>”
=> an exception occured when calling OS_BINDING
ERR SAS ”OS_BINDING error”
”Exception received in call to OS_BINDING”
ERR SAS ”Can not connect”
”Application name : <SAS_NAME>”
=> Application connected, but could not be registered internally (P_USER_REGISTER.CONNECT_SAS)
ERR SAS ”SAS is not loaded”
”Loaded application name : <SAS_NAME> Connecting application name : <SAS_NAME>”
ERR SAS ”Time out of Load_application”
”Application name : <SAS_NAME> did not connect within the the timeout period <NNNN> seconds (defined by the config variable TES.LOAD_APPLICATION_TIMEOUT).”
ERR SAS ”Time out of Load_application”
”Application name : <SAS_NAME> did not connect within the the timeout period <NNNN> seconds (defined by the config variable TES.LOAD_APPLICATION_TIMEOUT). Please note that manual loading of SASes is enabled.”
ERR SAS ”Connect_SAS called out of sequence”
”Application name : <SAS_NAME>”
ERR SAS ”Mode error”
”Connect_SAS called in illegal mode”
ERR SAS ”Mode error”

”Disconnect_SAS illegal in current mode”
ERR SAS ”Application not connected”
”Application ID: <NNNN>”
ERR SAS ”Mode error”
”Send_error_message illegal in current mode”
ERR SAS ”Mode error”
”Get_SAS_status_ID illegal in current mode”
ERR SAS ”Get SAS status ID”
”Not connected: <NAME>”
ERR SAS ”Get SAS status ID”
”Invalid name: <NAME>”

INFO SAS ”SAS disconnected due to stop of TES”
”<SAS_NAME>”

<XXX> SAS ”<SAS_TEXT1>”
”<SAS_TEXT2>”

TYPE GROUP Text

ERR TES ”STORAGE_ERROR raised!”
”A STORAGE_ERROR exception was raised in SW Unit <UNIT_NAME> .
Please adjust the storage size for the task affected”
ERR TES ”Exception raised!”
”Exception: <EXCEPTION_NAME> was raised in SW Unit: <UNIT_NAME>”
ERR TES ”Coded Message Error”
”The provided sid does not correspond to a user message definition.”
ERR TES ”AP Communicator Error”
”TES internal error : <EXCEPTION_NAME> occurred.”
ERR TES ”Initial setup failure.”
”TES internal error : <EXCEPTION_NAME> occurred.”
ERR TES ”Initial setup failure”
”TES internal error : <EXCEPTION_NAME> occurred.”
ERR TES ”AP Controller Error”
”TES internal error : <EXCEPTION_NAME> occurred.”
ERR TES ”Sending message failed”
”Return status from General Comms: <STATUS>”
ERR TES ”Initialisation Error”
”Invalid mode in the INIT operation.”
ERR TES ”Start Error”
”Need initialisation before START.”
ERR TES ”Start Error”
”Invalid mode in the START operation.”
ERR TES ”Stop Error”
”Invalid mode in the STOP operation.”
ERR TES ”TSS Error”
”<TEXT>”
Already member of another SMT–domain.
—> Has the TSS process on the test node already been initialised with
—> another configuration, defining a different SMT domain (i.e. e.g. having
—> a different MTP)

—> Stop whole test configuration and start again

_____________ Issuing node not SMT–server for given domain.
Return code = TS_OPS_TYPES.ETSNOTSMTSRV
—> was the TSS server (TSP) not setup correctly before ?
—> Setup the whole test configuration again using TSCV
—> If not successful: restart TSP process and try again

ERR TES ”Simulator received illegal command.”
””

ERR TES ”Calibrated value log failure”
”Logging of <ENG_VALUE_TYPE> items not supported, for <PATHNAME>.”

ERR TES ”Data type conversion failure”
”Type <ENG_TYPE> could not be converted, for <PATHNAME>”

ERR TES ”Data type conversion failure”
”Type <RAW_TYPE> could not be converted, for <PATHNAME>”

ERR TES ”Could not insert SID in local SID table”
”Not enough memory could be found to create the local SID table for data distribution”

ERR TES ”Error Message Loading Failure.”
”The status <STATUS> was received when loading coded error message definitions.”

ERR TES ”Announce enditems to HCI failed”
”Logging to HCI might fail but proceeding with setup”

ERR TES ”Init kernel failed”
”The return status was <STATUS>”

ERR TES ”Init Clock failed”
”Check if tss is running. The return status was <STATUS>”

ERR TES ”Init Archive failed”
”The return status was <STATUS>”

ERR TES ”Init UCLI failed”
”The return status was <STATUS>”

ERR TES ”Init CM failed”
”The return status was <STATUS>”

ERR TES ”Init GDU manager failed”
”The return status was <STATUS>”

ERR TES ”Init Simulator failed”
”The return status was <STATUS>”

ERR TES ”Init Replayer failed”
”The return status was <STATUS>”

ERR TES ”Error communicating to remote TES”
”cannot inform remote TES <TES_NAME> about completion of TC Verification”

ERR TES ”Error during setup of communication”
”Return status from ADT_SYSTEM_TOPOLOGY.WHO_AM_I: <STATUS>”

ERR TES ”Error during setup of communication”
”Return status from GENERAL_COMMS.SETUP_COMMUNICATION_CONFIGURATION: <STATUS>”

ERR TES ”Error reading message : <STATUS>”
”Error reading message : <STATUS>”

ERR TES ”Initialisation error.”
”Initialisation error. Simulator must be initialised in simulation–mode.”

ERR TES ”Could not stop Simulator properly.”

”TES internal error : Could not put ADU_Generator into ADU_Generator_Pool.”
ERR TES ”TES internal error.”
”TES internal error : Could not put ADU_Generator into ADU_Generator_Pool.”
ERR TES ”Simulation–data corrupted.”
”Simulation–data corrupted. Measurement <PATHNAME>) in ADU (<PATHNAME>) already exists.”
ERR TES ”Could not start simulate ADU.”
”Could not start simulate ADU <PATHNAME>. Could not get a new ADU_Generator from the ADU_Generator_Pool.”
ERR TES ”Could not stop simulation of ADU.”
”TES internal error : Could not stop ADU <PATHNAME> properly. Could not put ADU_Generator into ADU_Generator_Pool.”
ERR TES ”Initial setup failure.”
”TES internal error : Could not create datastructure.”
=> Error when creating a message buffer for AP/SAS messages
=> or Error when creating a buffer for APs (U_WORD_STORAGE)
=> ADT_SID_TO_GENERATOR_MAP.CREATE_ERROR
ERR TES ”Initialisation Error.”
”Could not initialise the PI Server of the TES Core.”
ERR TES ”Start Error.”
”Could not start the Clock manager.”
ERR TES ”Start Error.”
”Could not start the Archive manager.”
ERR TES ”Start Error.”
”Could not start the User manager.”
ERR TES ”Start Error.”
”Could not start the Kernel.”
ERR TES ”Start Error.”
”Could not start the GDU manager.”
ERR TES ”Start Error.”
”Could not start the Data Dispatch Service.”
ERR TES ”Start Error.”
”Could not start the UCL manager.”
ERR TES ”Start Error.”
”Could not start the SW Command manager.”
ERR TES ”Start Error.”
”Could not start the Simulation manager.”
ERR TES ”Start Error.”
”Could not start the Replayer.”
ERR TES ”Start Error.”
”Could not start the TES Core.”
ERR TES ”Start Error.”
”Could not start the PI Server of the TES Core.”
ERR TES ”Stop Error.”
”Could not stop the Simulation manager.”
ERR TES ”Stop Error.”
”Could not stop the Replayer.”
ERR TES ”Stop Error.”

”Could not stop the UCL manager.”
ERR TES ”Stop Error.”
”Could not stop the Data Dispatch Service.”
ERR TES ”Stop Error.”
”Could not stop the GDU manager.”
ERR TES ”Stop Error.”
”Could not stop the SW cmd manager.”
ERR TES ”Stop Error.”
”Could not stop the Kernel.”
ERR TES ”Stop Error.”
”Could not stop the User manager.”
ERR TES ”Stop Error.”
”Could not stop the Archive manager.”
ERR TES ”Stop Error.”
”Could not stop the Clock manager.”
ERR TES ”Stop Error.”
”Could not stop the PI Server of the TES Core.”
ERR TES ”Shutdown Error.”
”Could not shut down the PI Server of the TES Core.”
ERR TES ”Initialisation Error.”
”Initialisation only allowed in Available and Idle modes and
when shutdown is not in progress”
ERR TES ”Initialisation Error.”
”Initialisation when in Available mode must be with forced loading of MDB data.”
ERR TES ”Initialisation Error.”
”Wrong Active Mode provided in Init (Use Normal, Simulation or Replay).”
ERR TES ”Initialisation Error.”
”Can not find TES instance name in System Topology Table.”
ERR TES ”Initialisation Error.”
”Could not stop TES Core Idle mode.”
ERR TES ”Initialisation Error.”
”Could not initialise the TES Core.”
ERR TES ”Initialisation Error.”
”Could not start TES Core to Idle mode.”
ERR TES ”Start Error.”
”Start not allowed in current mode.”
ERR TES ”Stop Error.”
”Stop not allowed in current mode.”
ERR TES ”Stop Error.”
”Could not stop the TES Core (Fatal Error).”
ERR TES ”Stop Error.”
”Could not stop the TES Core.”
ERR TES ”Suspend Error.”
”Current mode is <MODE> . Suspend is only meaningful in SIMULATION and REPLAY mode.”
ERR TES ”Suspend Error.”
””
ERR TES ”Resume Error.”
”Current mode is <MODE>. Resume is only meaningful in SIMULATION and REPLAY mode.”

ERR TES ”Resume Error.”
”Could not resume simulation.”
ERR TES ”Resume Error.”
”Could not resume replaying.”
ERR TES ”Shutdown Error.”
”Shutdown not allowed in current mode.”
ERR TES ”Shutdown Error.”
”Could not stop the TES Core.”
ERR TES ”Synchronize Error.”
”Synchronize not allowed in current mode.”
ERR TES ”Synchronize Error.”
”Synchronization not possible for equal client and server.”
ERR TES ”Synchronize Error.”
”Could not synchronize with remote TES : <TES_NAME>”
ERR TES ”Connect error”
”Could not connect with remote TES : <TES_NAME>”
ERR TES ”Operation not Available”
”TES must be shutdown first!”
ERR TES ”Could not read HK–values from TSS.”
”<TEXT>”
ERR TES ”Wrong user calling”
”Only HCI and TSCV can call this operation, not : <NAME>”
ERR TES ”HCI/TSCV not connected”
”HCI/TSCV name : <NAME>”
ERR TES ”Name and ID conflict”
”Name: <NAME> and ID: <NNNN> are not connected in CM”
ERR TES ”Operation not allowed in <MODE> mode”
”Operation <OPERATION>”
ERR TES ”TES internal error.”
”TES internal error : No free AP info map entry.”
ERR TES ”Could not store message.”
”Could not store message from AP–ID <NNNN> to AP–ID <NNNN>. Not enough memory.”
ERR TES ”Could not store message.”
”Could not store message from Application with ID <NNNN> to AP–ID <NNNN>.
Not enough memory.”
ERR TES ”Get Message Error”
”TES internal error : <EXCEPTION_NAME> occurred.”

WRN TES ”CLOSE_ARCHIVE ignored”
”UCL system library routine CLOSE_ARCHIVE ignored in replay mode.”
WRN TES ”ENABLE_ARCHIVING ignored”
”UCL system library routine ENABLE_ARCHIVING ignored in replay mode.”
WRN TES ”DISABLE_ARCHIVING ignored”
”UCL system library routine DISABLE_ARCHIVING ignored in replay mode.”
WRN TES ”START_SMT ignored”
”UCL system library routine START_SMT ignored in replay mode.”
WRN TES ”STOP_SMT ignored”
”UCL system library routine STOP_SMT ignored in replay mode.”
WRN TES ”Trying to connect null name”

””

WRN TES ”Application cannot connect”
”Application <NAME> not connected, max users connected”

INFO TES ”Connected to Current CCU.”
””

INFO TES ”Connected to HCI.”
””

INFO TES ”Connection to HCI failed.”
”Status= <STATUS>”

INFO TES ”Disconnecting from HCI.”
””

INFO TES ”Connected to TRDB (DBS).”
””

INFO TES ”Disconnected from TRDB (DBS).”
””

INFO TES ”Stopped DBS RPI.”
””

INFO TES ”Disconnected from HCI.”
””

INFO TES ”Entering ERROR mode.”
””

INFO TES ”Entering AVAILABLE mode.”
””

INFO TES ”Initialisation parameters.”
”Mode: <MODE>
MTP: <TRUE/FALSE>
Load from DB: <TRUE/FALSE>
CCU: <PATHNAME>
Measurements: <NNNN>
ADUs: <NNNN>
GDUs: <NNNN>
GDU lists: <NNNN>
Mon. lists: <NNNN>
APs: <NNNN>
User libs: <NNNN>
EGSE nodes:<NNNN>
User MSGs: <NNNN>”

INFO TES ”Entering <MODE> mode.”
””

INFO TES ”Suspended simulation.”
”Suspended simulation on <NODE>”

INFO TES ”Resumed simulation.”
”Resumed simulation on <NODE>”

INFO TES ”Shutting down now ...”
””

INFO TES ”Application connected”
”Application name : <NAME>”

INFO TES ”User disconnected <SAS_NAME>”

””
INFO TES ”HCI/TSCV connected”
”Name : <NAME>. This user was already connected.”
INFO TES ”HCI/TSCV connected”
”Name : <NAME>”
INFO TES ”HCI/TSCV disconnected”
”Name : <NAME>”

MSG TES ”<TEXT1>”
”Trace: <TEXT2>”

TYPE GROUP Text

ERR TRDB ”Failed to log engineering value.”
”Logging of the engineering value <PATHNAME> failed with the status <STATUS>”
ERR TRDB ”Initialisation Error.”
”The status <STATUS> was received when connecting to TRDB (DBS).”
ERR TRDB ”Shutdown Error.”
”The status <STATUS> was received when disconnecting from TRDB (DBS).”
ERR TRDB ”Shutdown Error.”
”The status <STATUS> was received when stopping the DBS RPI.”

INFO TRDB ”EVL status changed.”
”The engineering value logging for <PATHNAME> is turned <ON/OFF>”

TYPE GROUP Text

ERR UCLI ”Module in Use”
”Attempted to overwrite an I–Code module which is in use. Automated Procedure : <PATHNAME>”
ERR UCLI ”Module in Use”
”Attempted to overwrite an I–Code module which is in use. User Library : <PATHNAME>”
ERR UCLI ”Automated Procedure not found”
”File for Automated Procedure <PATHNAME> could not be loaded.”
ERR UCLI ”I–code for Automated Procedure not found”
”Automated Procedure <PATHNAME> could not be loaded due to missing I–code (not compiled?)”
ERR UCLI ”User Library not found”
”File for User Library <PATHNAME> could not be loaded.”
ERR UCLI ”I–code for User Library not found”
”User Library <PATHNAME> could not be loaded due to missing I–code (not compiled?)”
ERR UCLI ”Internal error.”
”Time_Slicer: Removed too many clients.”
ERR UCLI ”AP initialisation failure.”
”Could not initialise AP <PATHNAME>”
ERR UCLI ”Initialisation error”
”Could not start AP: <PATHNAME>. User library <PATHNAME> could not be loaded”
ERR UCLI ”Initialisation error”
”Could not execute user library routine (invalid code).”
ERR UCLI ”Initialisation error”

”Could not start AP–ID <NN>; could not access AP I–Code.”
ERR UCLI ”Initialisation error”
”Could not start AP <PATHNAME>. An error occurred during loading of I–code into the stack machine data structures.””
ERR UCLI ”Initialisation error”
”Could not start AP : <PATHNAME>. <EXCEPTION_NAME> occurred.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Error during the execution of a system library routine.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Error during the execution of a system library routine.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. An unexpected exception occured during execution.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. An unexpected exception occured during execution.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. This instruction is illegal in DOUBLE mode.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. This instruction is illegal in DOUBLE mode.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. This instruction is illegal in MULTI mode.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. This instruction is illegal in MULTI mode.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Constraint– or numeric error raised.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Constraint– or numeric error raised.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. ~:~ (’no time’) used as a time operand.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. ~:~ (’no time’) used as a time operand.”

ERR UCLI ”TES Internal error : Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Illegal operator.”

ERR UCLI ”TES Internal error : Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Illegal operator.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Error when accessing the TES local database.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Error when accessing the TES local database.”

ERR UCLI ”TES Internal error : Execution error”
”Stopped execution of AP: <PATHNAME>. The current stack machine status is: <OK/ERROR/HALT>”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. This is an illegal instruction at this point.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. This is an illegal instruction at this point.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. A range or index check resulted in an error.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. A range or index check resulted in an error.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Stack underflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Stack underflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Stack overflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Stack overflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Memory underflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Memory underflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Memory overflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Memory overflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. The PC is outside the code frame.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. The PC is outside the code frame.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Error trap; the error code was: <NNNN>”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Error trap; the error code was: <NNNN>”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME>. FATAL : An exception occurred in the ERROR_TRAP routine.”

ERR UCLI ”Execution error”
”AP: <PATHNAME>. Could not store string read from the TES local database; not enough space was allocated for the string. SID: <NNNN>”

ERR UCLI ”Execution error”
”AP: <PATHNAME>. Could not write string to the TES local database (too long). SID: <NNNN>”

ERR UCLI ”Execution error”
”AP: <PATHNAME>, Instruction : <INSTRUCT>. The duration resulting from the subtraction could not be handled by the current Ada implementation.”

ERR UCLI ”Termination error”
”Could not terminate AP properly: <PATHNAME>. An error occurred when returning user library parameters to HCI.”

ERR UCLI ”Termination error”
”Could not terminate AP properly: <PATHNAME>. <EXCEPTION_NAME> occurred.”

ERR UCLI ”Could not open file for debug output”
”For Debug–Option ’<XXXXX>’. An exception occurred. Execution continues with debugging switched off.”

ERR UCLI ”Derived value error”
”<PATHNAME> : Error during the execution of a system library routine.”

ERR UCLI ”Derived value error”

”<PATHNAME> : An unexpected exception occured during execution.”
ERR UCLI ”Derived value error”
”<PATHNAME> : This instruction is illegal in DOUBLE mode.”
ERR UCLI ”Derived value error”
”<PATHNAME> : This instruction is illegal in MULTI mode.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Constraint– or numeric error raised.”
ERR UCLI ”Derived value error”
”<PATHNAME> :~:~ (’no time’) used as a time operand.”
ERR UCLI ”TES Internal error : Derived value error”
”<PATHNAME> : Illegal operator.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Error when accessing the TES local database.”
ERR UCLI ”TES Internal error : Derived value error”
”Stopped execution of AP: <PATHNAME>. The current stack machine status is: <OK/ERROR/HALT>”
ERR UCLI ”Derived value error”
”<PATHNAME> : illegal instruction.”
ERR UCLI ”Derived value error”
”<PATHNAME> : A range or index check resulted in an error.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Stack underflow.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Stack overflow.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Memory underflow.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Memory overflow.”
ERR UCLI ”Derived value error”
”<PATHNAME> : The PC is outside the code frame.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Error trap; the error code was: <NNNN>”
ERR UCLI ”Derived value error”
”<PATHNAME> : FATAL : An exception occurred in the ERROR_TRAP routine.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Could not store string read from the TES local database; not enough space was allocated for the string. SID: <NNNN>”
ERR UCLI ”Derived value error”
”<PATHNAME> : Could not write string to the TES local database (too long). SID: <NNNN>”
ERR UCLI ”Derived value error”
”<PATHNAME> : The duration resulting from the subtraction could not be handled by the current Ada implementation.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Could not start calculation”
ERR UCLI ”Error executing System Library routine”
”AP: <PATNAME>. System Library routine READ_MESSAGE_FROM_AP raised <EXCEPTION_NAME>.”
ERR UCLI ”Error executing System Library routine”

”AP: <PATNAME>. System Library routine READ_MESSAGE_FROM_APPLICATION raised <EXCEPTION_NAME>.”

ERR UCLI ”Error in SW command parameter list”
”Too many parameters”

ERR UCLI ”Error in SW command parameter list”
”An illegal parameter type was found : <TYPE>”

ERR UCLI ”Error in SW command parameter list”
”An ””in out”” parameter was found in a parameter list.”

ERR UCLI ”Error in SW command parameter list”
”Parameter list too large: doesn’t fit into a CCSDS packet.”

ERR UCLI ”Error in GDU parameter list”
”Too many parameters”

ERR UCLI ”Error in GDU parameter list”
”An illegal one word scalar parameter type was found: <TYPE>”

ERR UCLI ”Error in GDU parameter list”
”An illegal two word scalar parameter type was found: <TYPE>”

ERR UCLI ”Error in GDU parameter list”
”An illegal parameter type was found: <TYPE>”

ERR UCLI ”Error in GDU parameter list”
”An illegal/incompatible scalar parameter type was found: <TYPE>”

ERR UCLI ”Error executing System Library routine”
”Request from <NAME>. Call to unknown System Library with library number =<NNNN>”

ERR UCLI ”Error executing System Library routine”
”Request from <NAME>. Call to unknown System Library routine of the <LIBRARY_NAME>; unknown procedure number =<NNNN>”

ERR UCLI ”Error executing System Library routine”
”Calling the System Library routine <ROUTINE_NAME> is not allowed from HLCL.”

ERR UCLI ”Error executing System Library routine”
”Request from <NAME>. Call to System Library routine <ROUTINE_NAME> returned with status <NNNN>”

ERR UCLI ”Error executing System Library routine”
”Request from <NAME>. Call to System Library routine <ROUTINE_NAME> gave an unexpected exception”

ERR UCLI ”Error executing System Library routine”
”AP: <PATHNAME>. Call to unknown System Library with library number =<NNNN>”

ERR UCLI ”Error executing System Library routine”
”AP: <PATHNAME>. Call to unknown System Library routine of the <LIBRARY_NAME>; unknown procedure number =<NNNN>”

ERR UCLI ”Error executing System Library routine”
”AP: <PATHNAME>. Call to System Library routine <ROUTINE_NAME> returned with status <NNNN>”

ERR UCLI ”Error executing System Library routine”
”AP: <PATHNAME>. Call to System Library routine <ROUTINE_NAME> gave an unexpected exception”

ERR UCLI ”Error executing System Library routine”
”Calls to <LIBRARY_NAME> are not allowed for derived values.”

ERR UCLI ”Error executing System Library routine”
”SET_DEFAULT_WORKSTATION: DEFAULT_WORKSTATION is unchanged.

<HCI_NAME> is not a participated HCI in test configuration.”
ERR UCLI ”Error executing System Library routine”
”SET_CCSDS_APID: Invalid parameter range. The CCSDS application id is in range 0 .. 2047.”
ERR UCLI ”Internal Error”
”Couldn’t wait for AP because EXECUTE_AP status was <NNNN>”
ERR UCLI ”Internal Error”
”Error clearing data structures <PATHNAME>. <EXCEPTION_NAME> occurred.”
ERR UCLI ”HK Value Write Error”
”Error writing statement no.: <PATHNAME>. <EXCEPTION_NAME> occurred.”
ERR UCLI ”Error when calling HCI RPI”,
”Received the following return status when calling
HCI_RPI.Return_UCL_user_lib_parameters : <STATUS>. The HCI instance
addressed was <NAME>”
ERR UCLI ”UCLI is stopped”
”Could not start UCL user library routine. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not start <PATHNAME>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not start AP. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not synchronise with AP, ID <NNNN>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Can not terminate any APs. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not terminate AP, ID <NNNN>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not suspend AP, ID <NNNN>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not resume AP, ID <NNNN>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not return AP identifier for <PATHNAME>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not write message to AP–ID <NNNN>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not write message to <PATHNAME>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not return status of AP, ID <NNNN>. UCLI is stopped.”
ERR UCLI ”Could not start AP.”
”<PATHNAME> cannot be started because it has parameters but no parameter was provided”
ERR UCLI ”Could not start library routine.”
”Could not start UCL user library routine.The maximum number of APs is already running.”
ERR UCLI ”Could not start emergency AP.”
”Could not start <PATHNAME>. The maximum number of APs is already running.”
ERR UCLI ”Could not start AP.”
”Could not start <PATHNAME>. The maximum number of APs is already running.”
ERR UCLI ”AP not found.”
”Could not synchronise with AP. AP–ID <NNNN> not found.”
ERR UCLI ”AP not found.”

”Could not terminate AP, AP–ID <NNNN> not found.”

ERR UCLI ”Could not suspend AP.”
”Could not suspend <PATHNAME>. The AP has already been suspended on request.”

ERR UCLI ”AP not found.”
”Could not suspend AP, ID <NNNN>. AP not found.”

ERR UCLI ”Could not resume AP.”
”Could not resume <PATHNAME>. AP was not suspended on request.”

ERR UCLI ”AP not found.”
”Could not resume AP, ID <NNNN>. AP not found.”

ERR UCLI ”AP not found.”
”Could not write message to AP–ID <NNNN>; AP not found.”

ERR UCLI ”AP not found.”
”Could not write message to AP. <PATHNAME> not found.”

ERR UCLI ”Internal Error”
”Attempted to release the wrong Wait Table entry.”

ERR UCLI ”Failed to terminate AP(s).”
”Failed to release wait events: <NNNN> out of <NNNN> external waits within the time–out period.”

ERR UCLI ”Failed to terminate AP(s).”
”Failed to terminate AP(s). <NNNN> APs out of <NNNN> confirmed termination within the time–out period.”

WRN UCLI ”Could not stop.”
”Could not stop: APs are running. FORCED STOP required.”

INFO UCLI ”Emergency AP started with AP_ID: <NNNN>”
”<PATHNAME> started with <PRIORITY> priority”

INFO UCLI ”AP started with AP_ID: <NNNN>”
”<PATHNAME> started with <PRIORITY> priority”

INFO UCLI ”AP finished for AP_ID <NNNN>”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. HALT completion code: <SUCCESS/FAILURE> & <NNNN>”

INFO UCLI ”AP finished for AP_ID <NNNN>”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. HALT completion code: <SUCCESS/FAILURE> & <NNNN>”

INFO UCLI ”AP suspended for AP_ID <NNNN>”
”<PATHNAME> suspended.”

INFO UCLI ”AP resumed for AP_ID <NNNN>”
”<PATHNAME> resumed.”

INFO UCLI ”AP, ID <NNNN>, terminated via the TES RPI.”
”AP <PATHNAME> terminated on request from <NAME>.”

TYPE GROUP Text

MSG UEVT ”<AP_TEXT>”
”<AP_TEXT>”

D-3.3.3 Messages from DBS (Test Result DB)

Comms error in DBS_COMMS.SEND_TO_FA_WITH_ACKNOWLEDGE
SEND_MESSAGE returns NW_BADDSTID
—> Is the FA SAS installed ?

Comms error in DBS_COMMS.SEND_TO_FA_WITH_ACKNOWLEDGE
SEND_MESSAGE returns ETIMEDOUT
—> Is the FA SAS running ?

END_ERROR raised during LOCAL_PRINT_SERVICES.CHECK_PRINT

ERROR in CENTRAL_EXEC_DISPATCHER.MAIN_TASK.”,
MESSAGE TYPE = INIT_EXECUTION_SESSION
E_OTHERS_OF_EXCEPTION. Cause: Exception OTHERS raised. EXCEPTION_NAME = NOT_ENOUGH_PACKED_DATA. Consequence: Current operation fails. Recovery act: None. Debug info: None.

ERROR in CENTRAL_EXEC_DISPATCHER.MAIN_TASK
MESSAGE TYPE = STORE_EVENT_SESSION_FILE”,
E_OTHERS_OF_EXCEPTION. Cause: Exception OTHERS raised. EXCEPTION_NAME = FATAL_ERROR. Consequence: Current operation fails. Recovery act: None. Debug info: None.

ERROR in CEN_FILE_SERV.STORE_FILE
File size: 50750464, Free space: 46821376
E_FILE_SYSTEM_FULL. Cause: Not enough space. Consequence: STORE FILE fails. Recovery act: See DBS_ERR_102. Debug info: STORE_FILE in dbs_central_files_services.a

ERROR in CEN_LOG_DATA_MGMT.STORE_EVENT_FILE.
ADT_PACKED_FILE fails for DELETE on /testnode/cgs–test1/HCI_01.03–03–98_15:30:09”,
”E_STORE_EXECUTION_RESULT_FILE_16. Cause: ADT_PACKED_FILE error. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_central_log_data_management.a

ERROR in CEN_LOG_DATA_MGMT.STORE_EVENT_FILE.
Session PATRICK, File /testnode/cgs–test1/HCI_01.03–03–98_15:30:09:024.EVT.034
E_STORE_EXECUTION_RESULT_FILE_11. Cause: Cannot store EVENT file. Consequence: Current operation fails fails. Recovery act: See DBS_ERR_312. Debug info: STORE_EVENT_FILE in dbs_central_log_data_management.a

ERROR in CEN_LOG_DATA_MGMT.STORE_EVENT_FILE.”,
ADT_PACKED_FILE fails for DELETE on /testnode/cgs–test1/HCI_01.03–03–98_15:30:13
E_STORE_EXECUTION_RESULT_FILE_16. Cause: ADT_PACKED_FILE error. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_central_log_data_management.a

ERROR in CEN_LOG_DATA_MGMT.STORE_EVENT_FILE
”Session PATRICK, File /testnode/cgs–test1/HCI_01.03–03–98_15:30:13:120.EVT.035”,
”E_STORE_EXECUTION_RESULT_FILE_11. Cause: Cannot store EVENT file. Consequence: Current operation fails fails. Recovery act: See DBS_ERR_312. Debug info: STORE_EVENT_FILE in dbs_central_log_data_management.a

ERROR in CEN_LOG_DATA_MGMT.STORE_EVL_FILE.
APPEND_LOCAL_EVL_TO_CENTRAL returns DBS_INTERNAL_PROBLEM, Producer TES_01,

Session
E_STORE_EXECUTION_RESULT_FILE_07. Cause: Cannot store EVL file. Consequence: Current operation fails fails. Recovery act: See DBS_ERR_304. Debug info: STORE_EVL_FILE in dbs_central_log_data_management.a

ERROR in CEN_TAB_MANA INT_INSERT_EVT_FILE_INTO_TAB
E_INTERNAL_INSERT_EVENT_FILE_INTO_TABLE_01. Cause: Wrong file format (/testnode/csf_hp2/TES_01.24–02–98_15:27:00:768.EVT.165). Consequence: Current operation fails. Recovery act: See DBS_ERR_208. Debug info: INTERNAL_INSERT_EVENT_FILE_INTO_TABLE in dbs_ce

ERROR in CLDM.STO_EVL_FL.APPEND_LOC_TO_CEN.
E_OTHERS_OF_EXCEPTION. Cause: Exception OTHERS raised. EXCEPTION_NAME = constraint_error. Consequence: Current operation fails. Recovery act: None. Debug info: None.

ERROR in CLDM.STO_EVL_FL.APPEND_LOC_TO_CEN
Producer TES_01, Session UWE, File /testnode/csf_hp2/TES_01.24–02–98_15:26:57:97
E_STORE_EXECUTION_RESULT_FILE_09. Cause: Cannot store EVL file. Consequence: Current operation fails fails. Recovery act: See DBS_ERR_302. Debug info: STORE_EVL_FILE in dbs_central_log_data_management.a

ERROR in DB_BASIC_SERVICES.CONVERT_CONFIGURATION_FILE_KIND
Cannot convert to DATA_CATALOG_TYPES.T_CONFIGURATION_FILE_KIND
E_OTHERS_OF_EXCEPTION. Cause: Exception OTHERS raised. EXCEPTION_NAME = CONVERSION_ERROR. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_db_basic_services.pad

ERROR in DB_BASIC_SERVICES.ROLLBACK.
”SQL_ERROR raised –1041, ORA–01041: internal error. hostdef extension doesn’t exi”,
”E_ORACLE_PBM. Cause: Oracle problem. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_db_basic_services.pad

ERROR in GET_FILE_SIZE
”UNIX_CALLS.STAT returns ENOENT for /testnode/sivq–tn1//archive/TES_01132_1998080”,
”E_DBS_UNIX_SERVICES_01. Cause: Cannot get file size. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_unix_services.a

ERROR in INIT_LOCAL
”CONNECT_TO_VICOS_PRODUCT returns ECONNREFUSED”,
”E_INIT_LOCAL_02. Cause: Cannot connect to CENTRAL_EXEC_PRODUCT. Consequence: The application is not connected to Central DBS. Recovery act: None. Debug info: INIT_LOCAL in dbs_central_comms.a

ERROR in MASTER_ARCHIVE.GET_MA_CONFIG_REFS.
E_OTHERS_OF_EXCEPTION. Cause: Exception OTHERS raised. EXCEPTION_NAME = CONVERSION_ERROR. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_ma_file.pad

ERROR in MA_EXECUTION.GET_EXEC_NAMES_SEL_CRIT_INT
SQL_ERROR raised –3114, ORA–03114: not connected to ORACLE

E_ORACLE_PBM. Cause: Oracle problem. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_ma_execution.pad

ERROR in MASTER_ARCHIVE.ADD_MA_EVENT_FILE_REF
SQL_ERROR raised –3114, ORA–03114: not connected to ORACLE
E_ORACLE_PBM. Cause: Oracle problem. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_ma_logged_data_file.pad

ERROR in CEN_LOG_DATA_MGMT.STORE_EVENT_FILE
ADD_MA_EVENT_FILE_REF returns DBS_ORACLE_PROBLEM, Session DEFAULT_TEST_SESSION,E_STORE_EVENT_FILE. Cause: Adding event file ref fails. Consequence: Current operation fails. Recovery act: See DBS_ERR_311. Debug info: Prcedure STORE_EVENT_FILE of dbs_central_log_data_management.a

ERROR in ONLINE_REFERENCE.GET_NUMBER_OF_EVALUATION_USER.”,
SQL_ERROR raised –1092, ORA–01092: ORACLE instance terminated. Disconnection for
E_ORACLE_PBM. Cause: Oracle problem. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_online_reference.pad

NAME_ERROR raised during L_SW_VAR_A.GET_TRDB_INFO
(no supplement)

(every minute) TES_01 TRDB SEVERE DBS_INT_PROBL MSG_# 13
—> when HCI is running: Combination with msg:
Could not read HK–values from DBS_RPI
DBS HK–values will not be written. Received return status
DBS_UNIX_PROBLEM from DBS_RPI.GET_TRDB_INFO.
SW Unit: $RCSfile: p_dbs_hk_collector_.a,v $
—> The connection to Central DBS might be disturbed
—> Check if a file $DBS_HOME/data/adatmp<tmp> is created recently (minutes ago)
—> if not : raise SPR

ORACLE_INIT_FAILED in FA_LISTENER.MAIN_TASK
—>The central DBS processes had a problem to connect to Oracle
—> Is Oracle running ? Search for additional messages from DBS

ORACLE_INIT_FAILED in CENTRAL_EVAL_DISPATCHER.RUN_EVAL
—>The central DBS process serving the evaluation user had a problem to connect to Oracle
—> Is Oracle running ? Search for additional messages from DBS

ORACLE_INIT_FAILED in CENTRAL_EXEC_DISPATCHER.MAIN_TASK
—>The central DBS process serving the online test execution had a problem to connect to Oracle
—> Is Oracle running ? Search for additional messages from DBS

STATUS_ERROR raised during L_SW_VAR_A.GET_TRDB_INFO
—> ???

STORAGE_ERROR raised during LOCAL_PRINT_SERVICES.CHECK_PRINT
—> went out of memory
—> see below for workarounds on problems

–SQL_ERROR in DB_BASIC_SERVICES.CONNECT_TO_ORACLE
ORA–01034: ORACLE not available
—> The Oracle Processes are not running
—> Call your system administrator
—> The jprocesses are running on the DB Server node. They should be
—> available after boot of the OS.
—> They can also be started using the SQLDBA tool of Oracle.

SQL_ERROR in ONLINE_REFERENCE.GET_NUMBER_OF_EVALUATION_USER
ORA–03113: end–of–file on communication channel
SQL_ERROR in MA_EXECUTION.GET_EXEC_NAMES_SEL_CRIT_INT
ORA–03114: not connected to ORACLE

—> Are the Oracle Processes Running ?
—> Have the Oracle Prcesses been restarted when DBS was running
—> (i.e. are they started later than DBS ?)
—> go to Oracle / DB Server and verify via ps
—> Restart Oracle and/or DBS processes
—> Restart your application
—> (Note: TSCV will restart DBS automatically during startup)

SQL_ERROR in CEN_TAB_MANAG INSERT_EVT_BUFF_INTO_TABLE.
ORA–00942: table or view does not exist
INSERT_EVENTS_INTO_TABLE_PROBLEM in CEN_LOG_DATA_MGMT.STORE_EVENT_FILE.
ERR_311: Local EVENT not stored to Central.DEFAULT_TEST_SESSION,
/GSAF_HOME/dbs/data/WORK/DEFAULT_TEST_SESSION/TSCV....EVT.810>,
DBS_ORACLE_PROBLEM
—> ?? (user access problem ?? SPR ??)

SQL_ERROR in CEN_TAB_MANAG INSERT_EVT_BUFF_INTO_TABLE.
ORA–01653: unable to extend table space
—> The tablespace EVENT_SPACE could not be extended
—> Verify via sqlplus:
SQL > select * from user_free_space where tablespace_name = 'EVENT_SPACE';
—> Remove /archive old test sessions
—> Close actual test session by DBS recovery scripts, if still open
—> Restart DBS on DB Server

SQL_ERROR in CEN_TAB_MANAG INSERT_EVT_BUFF_INTO_TABLE.
ORA–01858: a non–numeric character was found where a numeric was expected
—> Something went wrong when inserting an event into the TRDB log
—> Write SPR

SQL_ERROR in DB_BASIC_SERVICES.CONNECT_TO_ORACLE
ORA–01034: ORACLE not available
—>The application tried to connect to DBS, which in turn tried to connect to Oracle. But Oracle

—> is not available
—> Check on DB_Server node with ps –auxw | grep ora
—> are all processes running: ora_pmon_<oraclesid>
—> ora_dbwr_<oraclesid>
—> ora_lgwr_<oraclesid>
—> ora_smon_<oraclesid>
—> orasrv
—> Startup these processes and restart the application

STOP_FA_SAS_REQUEST_FAILURE in FA_CONTROLLER.START_CONTROLLER(1).
DBS_ERR_806: Central Archive failed to send STOP request to Final Archive. 1 return DBS_COMMUNICATION_PROBLEM
—> Is the FA SAS active?

<u>* WARNING from CENTRAL_EXEC_DISPATCHER.MAIN_TASK</u>**.
PROCESS_STOP_REQUEST returns DBS_COMMUNICATION_PROBLEM
W_STOP_CENTRAL_03. Cause: Stop Request Refused. Consequence: Central DBS is not stopped. Recovery act: None. Debug info: PROCESS_STOP_REQUEST in dbs_central_exec_dispatcher.a

<u>* WARNING from L_CY_EVENT_SEND.CYCLIC_EVENT_SENDER</u>**
Error DBS_INTERNAL_PROBLEM occurred when Cyclic_Processing
—> Is Central DBS still running an OK ?

<u>* WARNING from L_CY_EVENT_SEND.CYCLIC_EVENT_SENDER</u>**
Error DBS_UNIX_PROBLEM occurred when Cyclic_Processing
—> There might be a problem accessing files / directories / disk
—> under $DBS_HOME/data/WORK

<u>* WARNING from L_CY_EVENT_SEND.CYCLIC_EVENT_SENDER.</u>**
”CYCLIC_EVENT_PROCESSING returns DBS_UNIX_PROBLEM”,
”W_CYCLIC_EVENT_PROCESSING. Cause: Internal error. Consequence: Events are possibly lost. Recovery act: None. Debug info: dbs_local_cyclic_event_sender.a

<u>WRONG_USER_NAME in LO_SE_MNGT.CONNECT_TRDB (DBS_INT_PROB)</u>
—> The user executing the tool which tried to connect to Central DBS is not
—> registered in $CGS_HOME/config/USER_PROFILES
—> the following entry is an example for user ’cgs_2’:
NAME=cgs_2
PRIV=IS_TEST_OPERATOR

Messages from Central DBS:

DBS_ERR_101 : The asynchronous request to remove Evaluation Result file failed :
Followed by : ”<’Filename’>.”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS failed to complete a deletion request. ”
” The Unix Error Number was reported in a preceding Error message. ”
RECOVERY_END:

DBS_ERR_102 : The asynchronous request to store a file into the TRDB failed
because the TRDB disk is full:
RECOVERY_START:
” Central DBS failed to complete store file request, Disk is full. ”
” – If the file concerned was a Raw Data file or an Execution ”
” Result file, a synchronous Status has been given back to the Requester, ”
” there is no reference to the file to store and no Inconsistency :”
” file is left at initial location without any reference in TRDB.”
” – If the file concerned was not a Raw Data file or an Execution ”
” Result file, a reference has been created for the file to store. ”
” file is left at initial location and is referenced into TRDB.”
” This applies for Evaluation Session files.”
” The file must be deleted using the Recovery Scripts ”
” <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_103 : The request to store a Raw Data file into the TRDB failed during addition of
Oracle references, an exception was raised:
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Storage of file into TRDB failed, Central DBS failed to add Raw Data”
” file references into ORACLE TRDB.”
” A synchronous Status has been given back to the Requester, ”
” there is no reference to the file to store and no Inconsistency:”
” file is left at initial location without any reference in TRDB.”
RECOVERY_END:

DBS_ERR_104 : The request to store an Execution Result file into the TRDB failed during
addition of Oracle references, an exception was raised:
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Storage of file into TRDB failed, Central DBS failed to add Execution Result”
” file references into ORACLE TRDB.”
” A synchronous Status has been given back to the Requester, ”
” there is no reference to the file to store and no Inconsistency:”
” file is left at initial location without any reference in TRDB.”
RECOVERY_END:

DBS_ERR_105 : Central DBS failed to return the acknowledge (following) to Local node during storage of Raw Data file or Exec Result file into the TRDB :
Preceeded by : ’failure status’
Followed by : ”’ACK to provide’ to ’Producer’ : <’Filename’>, ’Session’.”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” During storage of an Execution Result file or a Raw Data file,”
” Central DBS failed to return the status of the operation to the”
” requesting application.”
” This message could explain a DBS_COMMUNICATION_PROBLEM or
” COMMS_TIME_OUT status”
” obtained on a local node.”
RECOVERY_END:

DBS_ERR_106 : Central DBS failed to update file refs during asynchronous storage of file into the TRDB :
Followed by : ”<’Filename’>, ’Session’, ’Producer’, ’File type’.”
RECOVERY_START:
” Storage of a file into TRDB failed: Central DBS failed to update ”
” file references with the new location of the file.”
” The file is not deleted from its initial location, and its ORACLE”
” references are pointing to initial location. The file could be ”
” already copied into its final destination (under Central TRDB). ”
” The file should be deleted using the Recovery Scripts ”
” <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_107 : Central DBS failed to delete the file just stored into the TRDB from its initial location. It is stored into TRDB but stays on its initial location.
Followed by : ”<’Filename’>, ’Session’, ’Producer’, ’File type’.”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” File stored into TRDB but Central DBS failed to delete it ”
” from its initial location.”
” This problem does not lead to any inconsistency. The file”
” specified in the message should however be deleted manually.”
RECOVERY_END:

DBS_ERR_108 : Central DBS failed to copy file into TRDB directory structure.
Followed by : ”<’Filename’>, ’Session’, ’Producer’, ’File type’.”
RECOVERY_START:
” Store of file failed, Central DBS failed to copy/move the”
” file from its initial location into TRDB. The failure”
” reasons are detailed in a preceeding message (e.g. file name”
” given is not valid, read/write permissions not set correctly...)”
” An ORACLE reference has been created for the file to store, ”
” file is left at initial location and is referenced into TRDB.”
” This applies for Evaluation Session files.”

” The file should be deleted using the Recovery Scripts ”
” <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_109 : The file type (result type) given for a file to store into TRDB is not consistent (e.g. Exec Result file with result type set to DATA SET, non initialised file type...).
Followed by : ”<’Filename’>, ’Session’, ’Producer’.”
RECOVERY_START:
” Storage of Result file into TRDB failed; the Result type given is invalid ”
” (e.g. Execution Result file with result type set to DATA SET,...)
” An ORACLE reference has been created for the file to store. ”
” file is left at initial location and is referenced into TRDB.”
” The file should be deleted using the Recovery Scripts ”
” <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_110 : Central DBS failed to create Application subdirectory to store the Raw Data files created by that Application.
Followed by : ”<’subdirectory name’>, <’Raw Data file name’>, ’Session’.”
RECOVERY_START:
” Store of file failed, Central DBS failed to create ”
” the Application subdirectory to store the Raw Data files”
” into TRDB. The failure reasons are detailed in a preceeding ”
” message (e.g. directory structure is corrupted...)”
” An ORACLE reference has been created for the file to store. ”
” file is left at initial location and is referenced into TRDB.”
” The file reference should be deleted using the Recovery Scripts”
” <File Storage Failure> Menu, and the TRDB file structure should be checked.”
RECOVERY_END:

DBS_ERR_111 : Central DBS failed to access a UNIX file/directory or an ORACLE table when building list for automatic archiving. Auto–archiving processing is aborted.
Followed by : ”<’object_name’> ’object_type’”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Automatic Archiving failed: Central DBS failed to collect ”
” necessary information to decide on which files had to be archived ”
” to save space. This message is following another one describing ”
” the problem encountered (e.g. Oracle down, directory not accessible...). ”
” This message alone does not give an indication of inconsistency.”
RECOVERY_END:

DBS_ERR_112 : Auto–archiving succeeded but did not free enough space on disk to raise secure level.
Followed by : ”(’SECURE_SPACE’ %) on <’central home directory name’>”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”

” No Recovery Scripts action has to be executed following that message.”
” Automatic Archiving saved all available data of the session whose ARCHIVE_MODUS”
” is set but failed to reach the SAFE level specified in the dbs configuration”
” file (the Automatic Archiving is starting when disk space used reaches a CRITICAL ”
” level, it archives files of the session whose ARCHIVE_MODUS is set”
” until a SAFE level is reached). ”
” That means either the CRITICAL and SAFE level are not defined in accordance ”
” to the local configuration or the disk space is used mainly by other sessions.”
” This message alone does not give an indication of inconsistency, it”
” shows a non–optimum configuration for the Automatic Archiving functionnality.”
RECOVERY_END:

DBS_ERR_113 : Central Archive failed to archive all remaining file of a session,
The problem desciption follows.
Session is not archived.
Followed by : ”’Session Name’, ’problem desciption’”
RECOVERY_START:
” Automatic Archiving failed: Central DBS failed to archive remaining files ”
” of the session when its closure was requested.”
” The session is not archived and its state is kept to the value To_Be_Archived.”
” This message could be preceeded by another one giving a precise description”
” of the problem encountered.”
” Session must be recovered by the Recovery Scripts <Session is Used> Menu”
” but specific actions could be required according to preceeding messages.”
RECOVERY_END:

DBS_ERR_114 : ARCHIVING_ERROR :
Auto–archiving failed, Central Archive Failed to Archive the list
of selected files for a given reason (This message is a high level
one to warn the user of the consequences, it should follows others
describing the problem on Central Arch or on FA–SAS).
Followed by : ”<’Session Name’>, ’Status of problem’”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Automatic Archiving failed to save available data of the session whose ”
” ARCHIVE_MODUS is set.”
” This message follows others giving a precise description of the ”
” problem encountered.”
RECOVERY_END:

DBS_ERR_115 : INCONSISTENCY : The deletion of a file just archived
on FA for a given Session failed. This file is still
on the TRDB disk but its references are updated (ARCH + path)
The Session Record is not updated (To Be ARchived) and the
Session directories must be deleted.
There is an inconsistency in the TRDB.
Followed by : ”<’file Name’>”
RECOVERY_START:
” The deletion of a file just archived on FA medium for the given session”

” failed. File is archived on FA medium and still on the TRDB disk. It is”
” referenced as archived (on FA medium). The Session info is not updated”
” and is kept in the state To_Be_Archived.”
” The file has to be deleted from the TRDB disk and the Session must be”
” recovered using the Recovery Scripts <Session is Used> Menu.”
” This message is preceeded by another one giving a precise description”
” of the problem encountered (e.g. permission denied, NFS stale...).”
RECOVERY_END:

DBS_ERR_116 : INCONSISTENCY : The updating of references of a file just archived
on FA for a given Session failed. File is archived on FA medium
but its new location is not referenced in Oracle database
There is an inconsistency in the TRDB.
Followed by : ”<’fileName’>”
RECOVERY_START:
” The update of the reference for a file just archived on FA medium”
” failed. This message is preceeded by another one giving a precise description”
” of the problem encountered (e.g. Oracle down...).”
” A list of files has been archived on the FA medium; they are still on”
” the TRDB disk and referenced as so. The Session info is not updated”
” and is kept in the state To_Be_Archived. The Session must be”
” recovered using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_117 : INCONSISTENCY : The operation ARCHIVE_SESSION_ON_FA failed.
The update of a session status failed after this session data
has been archived on FA medium, the data references updated and
the data deleted from TRDB disk.
There is an inconsistency in the TRDB.
Followed by : ”’sessionName’”
RECOVERY_START:
” The operation ARCHIVE_SESSION_ON_FA failed. ”
” Central DBS successfuly archived the session files on the FA but it failed”
” to update the session status. This message could be preceeded by ”
” another one giving a precise description of the problem encountered ”
” (e.g. Oracle down...).”
” The Session info is not updated and is kept in the state To_Be_Archived.”
” The on–line data is also left to an out of date state.”
” The Session data must be recovered using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_118 : INCONSISTENCY : The operation ARCHIVE_SESSION_ON_FA failed.
DBS successfuly archived the files on the FA but it failed
to remove the directories of the session on the TRDB disk.
The File references are updated (ARC + new path).
The Session Record is not updated (To Be ARchived) and the
Session directories must be deleted.
There is an inconsistency in the TRDB.
Followed by : ”’Session Name’”
RECOVERY_START:

” The operation ARCHIVE_SESSION_ON_FA failed. ”
” Central DBS successfuly archived the session files on the FA,”
” updated the file references (archived) but it failed”
” to delete the Session data from the TRDB disk. This message is”
” preceeded by another one giving a precise description of the problem”
” encountered (e.g. Permission denied, NFS stale...).”
” The Session data must be recovered using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_119 : INCONSISTENCY_ERROR :
The operation IMPORT_SESSION_FROM_FA failed.
After DBS failed to communicate with FA–SAS OR FA–SAS failed to complete
Import of session from Final Archive disk, Central DBS tried to remove the
Session directory but failed.
There is an inconsistency in the TRDB as the New Session file structure
should be removed and the New Session record too (kept to the
state To_Be_IMported (SESSION_IS_USED)).
Followed by : ”’Export Name’, ’New Session Name’”
RECOVERY_START:
” The operation IMPORT_SESSION failed. ”
” Central DBS failed to import a session from an FA medium (see other”
” error/warning messages) and failed to remove the temporary file structure”
” created in prevision of the Import. This message is”
” preceeded by another one giving a precise description of the problem”
” encountered (e.g. Permission denied, NFS stale...).”
” The temporary file structure must be deleted from TRDB disk”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_120 : Central DBS failed to check automatic archiving option of the session;
if automatic archiving was specified for the given session, disc space
management is not executed following that error.
Followed by : ”<’fileName’>’, ’File Producer Node’,’sessionName’”
RECOVERY_START:
” Central DBS failed to check automatic archiving option of the ”
” given session during storage of a file; ”
” i.e. – Central DBS has not been able to read the ARCHIVE_MODUS of”
” session info in the ORACLE references OR”
” – Central DBS has not been able to read the free space available OR”
” – Central DBS has not been able to send an Archive request to the Central Archive process”
” Another message is preceeding describing precisely the error case.
” The result is that No Automatic Archiving processing can be ”
” executed (if required or not).”
” This message alone does not give an indication of inconsistency but”
” indicates a major problem that could lead to inconsistencies.”
RECOVERY_END:

DBS_ERR_121 : INCONSISTENCY : The update of online data table
failed after this session data is archived on FA medium and session state modified.
There is an inconsistency in the TRDB.

Followed by : ”’sessionName’”
RECOVERY_START:
” The operation ARCHIVE_SESSION_ON_FA failed. ”
” Central DBS successfuly archived the session files on the FA, ”
” updated the session status but it failed to update on–line data references.”
” This message could be preceeded by another one giving a precise description”
” of the problem encountered (e.g. Oracle down...).”
” The on–line data must be recovered using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_201 : INCONSISTENCY : The creation of a new Oracle Event Table for a given
Test Execution Session failed because this table already
existed. There is an inconsistency in the TRDB.
Followed by : ”<’Session Name’> ”
RECOVERY_START:
” The operation INITIALIZE_EVENT_TABLE failed. ”
” Central DBS detected that the event table it was supposed to create”
” already existed. There is only one event table per session and”
” that problem should not happen. It shows an inconsistency that can be ”
” related to other problems, e.g. deletion failure, invalid Oracle data base.”
” The event table should be deleted by the Recovery Scripts if no other”
” action is deduced from other inconsistencies (<Delete Session> Menu).”
RECOVERY_END:

DBS_ERR_202 : The Import of an Oracle Raw Data File Reference Table for a given
Test Execution Session failed because the file where this table has
been exported has a wrong format..
Followed by : ”<’File Name’, ’Session Name’> ”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of an Oracle Raw Data File Reference Table for the given”
” Test Execution Session failed because the file where this table has”
” been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_203 : The Import of an Oracle Config File Reference Table for a given
Test Execution Session failed because the file where this table has
been exported has a wrong format..
Followed by : ”<’File Name’, ’Session Name’> ”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of an Oracle Configuration File Reference Table for the given”
” Test Execution Session failed because the file where this table has”
” been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”

” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_204 : The Import of an Oracle Session Reference Table for a given
Test Session failed because the file where this table has
been exported has a wrong format..
Followed by : ”<’File Name’, ’Session Name’> ”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of the Oracle Session Reference Table for the given”
” Test Execution Session failed because the file where this table has”
” been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_205 : The deletion of an event table failed with Time out. Central DBS failed
to lock the table to remove it. After a given number of trial, it
interrupted the operation.
Followed by : ”’Session Name’, ’table name’”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of the Oracle Session Reference Table for the given”
” Test Execution Session failed because the file where this table has”
” been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_206 : The operation MANAGE_AA_SESSION_EVENT_SPACE failed : this operation
aims to move the content of an Oracle Event Table into a file referenced in the TRDB and
to archive it. The Table content has been copied into a given file and the Table references
have been updated with this file name, but then, the deletion of the event
table failed. The table must be deleted by the recovery scripts
Preceeded by : ’failure status’,
Followed by : ”’Session Name’”
RECOVERY_START:
” The deletion of an event table failed with Time out. Central DBS failed”
” to lock the table to remove it. After a given number of trial, it”
” interrupted the operation. It means that an Evaluator user is continuously”
” using the table. Check that no Local Node is blocked in an Evaluation operation”
” concerning events of that session.”
” This message is probably accompanied by others describing the higher level”
” implications of the problem; these could imply corrective actions.”
RECOVERY_END:

DBS_ERR_207 : The number of rows ’fetched’ by oracle does not correspond to the
expected number.
RECOVERY_START:

” The number of rows processed by ORACLE does not fit to the requested one.”
” The operation DELETE_DEFAULT_SESSION encountered a problem while”
” deleting DEFAULT_TEST_SESSION events. The variable ORACLE.SQLROWS”
” giving the number of rows processed by ORACLE has a value different of the number”
” of elements to delete. Refer to ORACLE documentation and Support.”
” This message is accompanied by others describing the higher level”
” implications of the problem; these should imply corrective actions.”
RECOVERY_END:

DBS_ERR_208 : Import of a session from the Final Archive failed.
The file referenced as Export file for the Session Events has not
the correct format.
Followed by : ”’File name’, ’Table Name’”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of the Oracle Event Table for the given Test”
” Execution Session failed because the file where this table has”
” been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_209 : The Import of an Oracle Result File Reference Table for a given
Session failed because the file where this table has been exported
has a wrong format.
Followed by : ”<’File Name’, ’Session Name’> ”
RECOVERY_START:
” The Import of a Session from FA medium failed.
” The Import of the Oracle Table containing Result file references”
” for the given Session failed because the file where this table has”
” been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_211 : Import of a session from the Final Archive failed.
The file referenced as Export file for the Session Event Table References has not
the correct format.
Followed by : ”’File name’, ’Table Name’”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of the Oracle Event Table for the given Test”
” Execution Session failed because the file where this table has”
” been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_212 : Import of a session from the Final Archive failed.
The file containing the EVL Short Identifiers is not found.
Followed by : ”’File name’, ’New Session Name’.”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of the Oracle Table containing Eng. Value Short Identifier references”
” for the given Test Execution Session failed because the ”
” file where these data has been exported is not found.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_213 : Import of a session from the Final Archive failed.
The file referenced as Export file for the Session Engineering Value File References has not the correct format.
Followed by : ”’File name’, ’New Session Name’”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of the Oracle Table containing Eng. Value file references”
” for the given Test Execution Session failed because the ”
” file where this table has been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_214 : Import of a session from the Final Archive failed.
The file referenced as Export file for the Session Engineering Value Short Identifiers has not the correct format.
Followed by : ”’File name’, ’New Session Name’”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of the Engineering Value Short Identifiers”
” for the given Test Execution Session failed because the ”
” file where this data has been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_215 : Management of Event space for Automatic Archiving failed.
Central DBS has not been able to create a new Event Oracle table for the session after it archived the current one to save space. The status of the operation is part of the message.
Preceeded by : ’failure status’,
Followed by : ”’Session Name’”
RECOVERY_START:
” The Management of Event space for Automatic Archiving failed.
” Central DBS detected a need for the Automatic archiving of the Events”

” of the session. It has archived the current Event Table to save space and”
” then has not been able to create a new empty Event Oracle table for”
” the given Session. The failure status is part of the message.”
” Other messages are related.”
” The session should be deleted using the recovery scripts <Delete Session> Menu.”
RECOVERY_END:

DBS_ERR_216 : Management of Event space for Automatic Archiving failed.
Central DBS has not been able to archive the current Event Oracle table
to save space. The status of the operation is part of the message.
Preceeded by : ’failure status’,
Followed by : ”’Session Name’”
RECOVERY_START:
” The Management of Event space for Automatic Archiving failed.
” Central DBS detected a need for the Automatic archiving of the Events”
” of the session. It has not been able to archive the current Event Oracle table
” to save space. The failure status is part of the message.”
” Other messages are related.”
” This message could have important consequences on the session data integrity if”
” the Event ORACLE Table Space is full and no action is taken to allow archiving.”
RECOVERY_END:

DBS_ERR_217 : Management of Event space for Automatic Archiving failed.
Central DBS has not been able to archive the current Event Oracle table
to save space.
RECOVERY_START:
” The Management of Event space for Automatic Archiving failed.
” Central DBS detected a need for the Automatic archiving of the Events”
” of the session. It has not been able to archive the current Event Oracle table
” to save space. An unexpected problem occured.”
” The failure status is part of the message.”
” This message could have important consequences on the session data integrity if”
” the Event ORACLE Table Space is full and no action is taken to allow archiving.”
RECOVERY_END:

DBS_ERR_218 : Management of Event space for Automatic Archiving failed.
Central DBS has not been able to get the Event Space free space to
check if Automatic archiving of Session Event table is required.
The status of the operation is part of the message.
Preceeded by : ’failure status’,
Followed by : ”’Session Name’”
RECOVERY_START:
” The Management of Event space for Automatic Archiving failed.
” Central DBS detected a need for the Automatic archiving of the Events”
” of the session. It has not been able to archive the current Event Oracle table
” to save space. The failure status is part of the message.”
” Other messages are related.”
” This message could have important consequences on the session data integrity if”
” the Event ORACLE Table Space is full and no action is taken to allow archiving.”
RECOVERY_END:

DBS_ERR_219 : INCONSISTENCY : The deletion of Default Session Events brougth an inconsistency into the TRDB: after having deleted the Events selected, Central DBS failed to update the reference of the Event table (time frame refs).
Preceeded by : ’failure status’,
Followed by : ”’new begin date’”
RECOVERY_START:
” The deletion of Default Session Events brougth an inconsistency into the TRDB:
” after having deleted the Events selected, Central DBS failed to update the ”
” reference of the Event table (time frame LT and SMT).”
” The DEFAULT_TEST_SESSION must be recovered using the <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_220 : Retrieve of Events from FA: Events were retrieved from the Final archive into an Oracle Table; the references concerned were updated but Central DBS failed to remove an Event file used for the retrieving. A warning showing the file name is issued to allow the user to delete it manually.
Followed by : ”’file name’, ’session name’”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” Events were retrieved from the Final archive into an Oracle Table. ”
” The references concerned were updated but Central DBS failed to remove the temporary”
” Event file used for the retrieving.”
” This problem does not lead to any inconsistency. The file”
” specified in the message should however be deleted manually.”
RECOVERY_END:

DBS_ERR_221 : The operation STORE_EVENT_TABLE failed : this operation aims to move the content of an Oracle Event Table into a file referenced in the TRDB. The Table content has been copied into a given temporary file and the Table references have been updated with this file name, but then, the storage of this temporary file into TRDB failed. The Event table is not deleted.
Preceeded by : ’failure status’,
Followed by : ”’Session Name’, ’file name’”
RECOVERY_START:
” Storage of a file into TRDB failed. The operation STORE_EVENT_TABLE failed.”
” This operation aims to move the content of an Oracle Event Table into a ”
” file referenced in the TRDB. The Table content has been copied into a given ”
” temporary file and the Table references have been updated with this file name.”
” After that, the storage of this temporary file into TRDB failed.”
” The Event table is not deleted.”
” The file should be deleted and the Oracle table references updated,”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_301 : Creation of a new central EVL file for local EVL file storage failed. A file and its references are created but not usable as central EVL file. The file may be unmoved to TRDB, or its references are not updated after being moved to TRDB.
Followed by : ”<file name without path>”
RECOVERY_START:

” Storage of an Eng. Value file into the TRDB failed.”
” The temporary file should be deleted and the Oracle table references updated, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_302 : Storage of an Eng. Value File from a Local Node into
the TRDB failed. Only a part of the Local Eng. Value File
is stored into the TRDB.
Followed by : ”’Session Name’, ’Local Node’, ’Local Eng. Value File Name’, ’Central Eng. Value File Name’”
RECOVERY_START:
” Storage of an Eng. Value file into the TRDB failed. Only a part of the Local Eng. Value ”
” file is stored into the TRDB.”
” The temporary file should be deleted and the Oracle table references updated, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_303 : Storage of an Eng. Value File from a Local Node into
the TRDB failed. No data is stored into TRDB.
Followed by : ”’Session Name’, ’Local Node’, ’Local Eng. Value File Name’”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Storage of an Eng. Value file into the TRDB failed. No data from the Local Eng. Value ”
” file is stored into the TRDB.”
” No reference for the local file has been created.”
RECOVERY_END:

DBS_ERR_304 : Storage of an Eng. Value File from a Local Node into
the TRDB failed. Central DBS has been able to create
an Oracle reference for that Local Eng. Value file.
Followed by : ”’Session Name’, ’Local Node’, ’Local Eng. Value File Name’”
RECOVERY_START:
” Storage of an Eng. Value file into the TRDB failed. ”
” Central DBS has been able to create an Oracle reference for that Local Eng. Value file.”
” The temporary file should be deleted and the Oracle table references updated, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_305 : Storage of an Eng. Value File from a Local Node into
the TRDB failed. Central DBS has been able store the Eng. Values
into TRDB but failed to update the Oracle references.
Followed by : ”’Central Eng. Value File Name’, ’Session Name’, ’Local Node’”
RECOVERY_START:
” Storage of an EVL file into the TRDB failed. ”
” Central DBS has been able to create an Oracle reference for that Local Eng. Value file.”
” It failed to update the Oracle reference.”
” The temporary file should be deleted and the Oracle table references updated, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_306 : Storage of an Eng. Value File from a Local Node into
the TRDB successful but Central DBS failed to delete the Local file.
Followed by : ”’Session Name’, ’Local Node’, ’Local Eng. Value File Name’”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” Storage of an EVL file into the TRDB is successful (data and Oracle references), ”
” but Central DBS failed to delete the Local file from its initial location.”
” This problem does not lead to any inconsistency. The file”
” specified in the message should however be deleted manually.”
RECOVERY_END:

DBS_ERR_307 :
RECOVERY_START:
” Storage of an EVL file into the TRDB is successful (data and Oracle references), ”
” but Central DBS failed to delete the Local file references in Oracle.”
” The Oracle table references must be updated using the DBS Recovery ”
” Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_308 :EVL stored to Central, but Central EVL file size check failed. May be too big, and still with ’CNA’ status. ”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
”This message is following others describing the problem with more details”
” Storage of an EVL file into the TRDB is successful (data and Oracle references), ”
” but Central DBS failed to check the size of the Central EVL file appended or”
” to update its reference to Central Accessible.”
” This message shows a general problem that could have future consequences”
” (e.g. another Oracle update failure or another problem accessing unix flat files)
” This problem could lead to others and the related messages should be analysed.”
RECOVERY_END:

DBS_ERR_309 : EVL stored to Central, but auto–archiving check failed. May be not enough
disc space.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” Automatic Archiving failed: Central DBS has not been able to check the ”
” available space or to process with the Automatic Archiving if required. ”
” This message could be preceeded by another one giving a precise description”
” of the problem encountered (Comms problem, Unix path not accessible...).”
” Consequently, the free space on the TRDB disk could be under the”
” expected limit. This problem could lead to others and the related messages ”
” should be analysed. Specific actions could be required according to”
” preceeding messages.”
RECOVERY_END:

DBS_ERR_311 : Local Event file not stored into TRDB
RECOVERY_START:
” Storage of an Event file into the TRDB failed. ”
” Central DBS has not been able to create an Oracle reference for this Event file or”
” it failed to insert the file content into the Oracle event table.”

” The temporary file should be deleted and the Oracle table references updated, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_312 : EVENTs from local file stored into TRDB, but local file not deleted
RECOVERY_START:
” Storage of an Event file into the TRDB failed. ”
” Central DBS has been able to create an Oracle reference for this Event file and”
” to insert the file content into the Oracle event table. But it failed to delete”
” the file. The local file and its reference remain.
” The temporary file should be deleted and the Oracle table references updated, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_313 : EVENTs from local file stored into TRDB, local file deleted, but not its Oracle reference
RECOVERY_START:
” Storage of an Event file into the TRDB failed. ”
” Central DBS has been able to create an Oracle reference for that Event file and”
” to insert the file content into the Oracle event table. It has deleted”
” the file but failed to remove its reference.
” The temporary Oracle table references must be removed, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_314 :EVENTs from local file not stored into TRDB due to unexpected problem
RECOVERY_START:
” Storage of an Event file into the TRDB failed due to an unexpected problem. ”
” The temporary file should be deleted and the Oracle table references updated, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_315 : Central failed to supply ’GET_EVL_LIST’ Acknowledge
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS encoutered a problem trying to send acknowledge of the given”
” operation to the caller. The problem could be described in a preceeding message.”
” This message could explain a DBS_COMMUNICATION_PROBLEM or COMMS_TIME_OUT status”
” given on a local node.”
RECOVERY_END:

DBS_ERR_316 : Eng. Values from local file not stored into TRDB due to unexpected problem
RECOVERY_START:
” Storage of an Eng. Value file into the TRDB failed due to an unexpected problem. ”
” The temporary file should be deleted and the Oracle table references updated, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_317 : During Session closure, DBS failed to store a file whose name is not formatted as an EVL or EVENT file”

” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS encoutered a problem while closing a session (session name is given)”
” The WORK directory for that session contains a file whose name is not formatted”
” as an EVL or EVENT file. This file is not expected at that location.”
RECOVERY_END:

DBS_ERR_401 : EVENT_TABLE_NOT_REMOVED
During initialisation of execution session, the Commit
failed with a fatal error.
Participating Applications Flag and Session dir space
(including stored Config files) has been removed; The FA
Auto Arch reservation are released.
Event Table can’t be removed after the fatal error.
Followed by : ”’execution session name’”.
RECOVERY_START:
” During initialisation of execution session, ORACLE failed to commit”
” the new session references.”
” DBS tried to recover partially by :”
” – resetting the Participating Applications Flags (to warn Participating Applications”
” of a session opening/closure), ”
” – removing the session directory structure (including stored Config files), ”
” – releasing the FA SAS device reservation (in case of AUTO ARCH selected). ”
” The Event Table created for the new session cannot be deleted after ORACLE”
” generated a fatal error.”
” The new session Event Table should be deleted by the Recovery Scripts as”
” soon as the ORACLE data base is in good state. <Delete Session> Menu”
RECOVERY_END:

DBS_ERR_403 : FILE_DELETION_FAILURE
During deletion procedure of default EVL, RD or RESULT files,
a problem occured while deleting one file.
Followed by : ”’file name’”.
RECOVERY_START:
” The operation Delete Default Session failed. ”
” A problem described in a preceeding message occured while deleting a”
” DEFAULT_TEST_SESSION file. ”
” The DEFAULT_TEST_SESSION is kept in the state To_Be_Partially_Deleted,”
” some references could be out of date. It should be recovered ”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_404 : DIR_DELETION_FAILURE (warning)
During execution session export, the temporary export
directory created by operation can’t be deleted.
Followed by : ”’dir path’”.
RECOVERY_START:
” Central DBS exported successfully an Execution session”
” onto an FA medium; but it failed to Remove the temporary Export ”
” directory structure created for that purpose.”

” The temporary Export directory must be deleted using the Recovery Scripts”
” <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_405 : SESSION_DIR_SPACE_NOT_REMOVED (warning)
During execution session deletion, the session info are
updated on ORACLE but session dir space is not removed.
Followed by : ”’session name’”.
RECOVERY_START:
” Central DBS successfully updated the ORACLE references”
” for a DELETE_EXECUTION_SESSION operation,”
” but it failed to Remove the session directory from TRDB disk.”
” The session directory must be deleted using the Recovery Scripts <Delete Session> Menu”
RECOVERY_END:

DBS_ERR_406 : SESSION_DELETION_FAILURE (warning/error)
During execution session deletion, an exception occured.
The session is not deleted and may be in a inconsistent state.
Followed by : ”’session name’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the DELETE_EXECUTION_SESSION operation.”
” This message is preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Delete Session> Menu.”
RECOVERY_END:

DBS_ERR_407 : DATA_DELETION_FAILURE (warning/error)
During retrieved data deletion, an exception occured.
The data may be not deleted and session may be in an inconsistent state.
Followed by : ”’session name’, ’data to delete’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the DELETE_RETRIEVED_EXECUTION_DATA operation.”
” This message is preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts.”
” The only way to recover is to delete all the on–line (retrieved) data”
” from the TRDB disk. The data can further be RETRIEVED from FA medium.”
” Use the <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_408 : DEFAULT_SESSION_DELETION_FAILURE (warning/error)
During default session deletion, an exception occured.
The data may be not deleted and session may be in an inconsistent state.
Followed by : –/–.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the DELETE_DEFAULT_TEST_SESSION operation.”
” This message is preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”

”   using the Recovery Scripts <Delete Session> Menu.”
RECOVERY_END:

DBS_ERR_409 :   INVALID_SESSION_STATE
The operation failed : Current execution session state
is not the attended one.
Followed by : ”’session name’, <’invalid session state’>”.
RECOVERY_START:
”   This error message alone does not describe an inconsistency in the TRDB.”
”   No Recovery Scripts action has to be executed following that message.”
”   Central DBS found an invalid value for an Execution Session State.”
”   It means that Local Nodes called a particular combination of operations”
”   (e.g. ARCHIVE_EXECUTION_SESSION and DELETE_EXECUTION_SESSION) at the ”
”   same time and that they both reset the Execution Session State without detecting the ”
”   other request. This should not lead to any inconsistency. ”
”   Note other messages related to the same session. ”
RECOVERY_END:

DBS_ERR_410 :   ARCHIVE_SESSION_FAILURE (warning/error)
During execution session archiving, an exception occured.
The session may be not archived and session may be in an inconsistent state.
Followed by : ”’session name’”.
RECOVERY_START:
”   Central DBS encoutered a problem during the execution of”
”   the ARCHIVE_EXECUTION_SESSION operation.”
”   This message is preceeded by another one describing the problem.”
”   The session is left in an inconsistent state and should be recovered”
”   using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_411 :   RETRIEVE_SESSION_FAILURE (warning/error)
During execution session retrieving, an exception occured.
The data may be not retrieved and session may be in an inconsistent state.
Followed by : ”’session name’”.
RECOVERY_START:
”   Central DBS encoutered a problem during the execution of”
”   the RETRIEVE_EXECUTION_SESSION operation.”
”   This message is preceeded by another one describing the problem.”
”   The session is left in an inconsistent state and should be recovered”
”   using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_412 :   EXPORT_SESSION_FAILURE (warning/error)
During execution session export, an exception occured.
The session is not successfully exported.
Followed by : ”’session name’”.
RECOVERY_START:
”   Central DBS encoutered a problem during the execution of”
”   the EXPORT_EXECUTION_SESSION operation.”
”   This message is preceeded by another one describing the problem.”
”   The session is left in an inconsistent state and should be recovered”

” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_413 : IMPORT_SESSION_FAILURE (warning/error)
During execution session export, an exception occured.
The session is not successfully imported and may be in an inconsistent state.
Followed by : ”’session name’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the IMPORT_EXECUTION_SESSION operation.”
” This message is preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_414 : EXPORT_PREPARATION_FAILURE (warning/error)
During execution session export, export of tables to files failed.
The session export is aborted.
Followed by : ”’error status’, ’session name’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the EXPORT_EXECUTION_SESSION operation. Export of tables to files failed.”
” This message can be preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_415 : TMP_DIR_DELETION_FAILURE (warning/error)
During execution session import, the temporary directory (EXPORT)
can’t be deleted after completion of table import operation.
The session import is aborted but imported files may be present on disc.
Followed by : ”’export dir path’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the IMPORT_EXECUTION_SESSION operation. The given temporary directory”
” cannot be deleted.”
” This message can be preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_416 : –/–
The closure of the Execution Session failed: Central
DBS failed to remove the directory structure warning the
local nodes of the session opening/closure.
The Execution Session is not closed.
Followed by : ”’Session name’, ’problem status’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the CLOSE_EXECUTION_SESSION operation. It failed to remove the directory structure”
” warning the participating applications of the session opening/closure.”

” The Execution Session is not closed.”
” This message can be preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Close Session> Menu.”
RECOVERY_END:

DBS_ERR_417 : SUPPLY_ACK_FAILURE
The communication failed: Central DBS failed to send
acknowledge of the operation to the caller.
Followed by : ”’operation acknowledge’, ’communication error status’”.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS encoutered a problem trying to send acknowledge of the given”
” operation to the caller. The problem could be described in a preceeding message.”
” This message could explain a DBS_COMMUNICATION_PROBLEM or COMMS_TIME_OUT
” status given on a local node.”
RECOVERY_END:

DBS_ERR_418 : UPDATE_FAILURE_WITH_INCONSISTENCY
The retrieving of files from FA succeed.
The retrieveing of the Event Oracle table succeed.
Files and Event Oracle table are present
on TRDB but update of Oracle File or Session references failed.
Followed by : ”’session name’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the RETRIEVE_EXECUTION_SESSION operation. The retrieving of files from FA succeeded.”
” The retrieveing of the Event Oracle table succeeded.”
” Files and Event Oracle table are present on TRDB but update of Oracle File or Session”
” references failed.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_419 : SESSION_STATE_FAILURE
During the export operation, the session state cannot be set back
to previous state given as argument. This is a severe error because
Session could be locked in a state where all action is forbidden.
Followed by : ”’previous session state’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the EXPORT_EXECUTION_SESSION operation. The session state cannot be set back”
” to previous state given as argument.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_420 : –/–
The closure of the Execution Session failed: Central
DBS failed to update EVL files Oracle refes.

The Execution Session is not closed.
Followed by : ”’Session name’, ’problem status’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the CLOSE_EXECUTION_SESSION operation. Central DBS failed update EVL”
” files Oracle references.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Delete Session> Menu.”
RECOVERY_END:

DBS_ERR_421 : SEND_WITHOUT_ACK_FAILURE
The closure of the Execution Session failed: Central Exec
failed to request Automatic Archiving of Session Remaining
files to Central Archive. The session is referenced as closed,
its Event table is stored into a file (refs commited)
The Execution Session is closed but not archive.
Followed by : ”’problem status’, ’Session Name’”.
RECOVERY_START:
” Automatic Archiving: Central DBS encoutered a problem during the execution of”
” the CLOSE_EXECUTION_SESSION operation. Central exec failed to send”
” a request to Central arch to archive the remaining data of the session.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Delete Session> Menu.”
RECOVERY_END:

DBS_ERR_422 : TMP_EVT_FILE_LEFT
During Archiving of an execution session, an error has been detected
The copy of session files on the FA_MEDIUM failed.
Central DBS tried to recover to the initial state of
the Session, it reset the Session State to Closed,
and tried to remove the temporary Event file containing
Session Events. This deletion failed File should be
deleted manually.
Followed by : ”’filename’, ’Session Name’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the ARCHIVE_EXECUTION_SESSION operation. The copy of session files on”
” the FA_MEDIUM failed. Central DBS tried to recover to the initial state of”
” the session: it reset the Session State to Closed, tried to remove the temporary”
” Event file containing Session Events. This deletion failed.”
” The given file must be deleted manually.”
RECOVERY_END:

DBS_ERR_423 : REMOVE_EVENT_TABLE_FAILURE
The archiving of an execution session was successful
(refs updated, files moved) until the last operation.
The dropping of the event table failed, it should be
removed by the recovery scripts
Followed by : ”’Session Name’”.
RECOVERY_START:

” Central DBS encoutered a problem during the execution of”
” the ARCHIVE_EXECUTION_SESSION operation. Central DBS failed to remove the Oracle”
” Event table after completion of the transfer to FA Medium.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Delete Session> Menu.””
RECOVERY_END:

DBS_ERR_501 : After having succesfully exported the Session on the
Final Archive device, Central DBS failed to remove the
temporary directory were it prepared data for Export.
This directory must be deleted manually.
This context message follows another one giving the Unix
problem details:
Followed by : ”’temporary directory name’”.
RECOVERY_START:
” Central DBS exported successfully an evaluation session”
” onto an FA medium; but it failed to remove the temporary export ”
” directory structure created for that purpose.”
” This context message follows another one giving the Unix problem details.”
” The temporary export directory must be deleted using the Recovery Scripts”
” <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_502 : After having succesfully imported the Session from the
Final Archive device, Central DBS failed to remove the
temporary directory were it prepared data for Import.
This directory must be deleted manually.
This context message follows another one giving the Unix
problem details:
Followed by : ”’temporary directory name’”.
RECOVERY_START:
” After having succesfully imported the session from the Final Archive device,”
” Central DBS failed to remove the temporary directory were it prepared data for Import. ”
” This context message follows another one giving the Unix problem details.”
” The temporary import directory must be deleted using the Recovery Scripts”
” <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_503 : The operation DELETE_EVALUATION_SESSION failed because the
Session state is not set correctly. The Local DBS should have
set it to the value ’To Be Partially Deleted’ or ’To Be Completely
Deleted’. Then it is an Internal Problem:
Followed by : ”’Session Name’, ’Session State’”.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS found an invalid value for an Evaluation Session State.”
” It means that Local Nodes called a particular combination of operations”
” (e.g. ARCHIVE and DELETE) at the same time and that they both reset”
” the Execution Session State without detecting the ”

” other request. This should not lead to any inconsistency. ”
” Note other messages related to the same session. ”
RECOVERY_END:

DBS_ERR_504 : Central DBS failed to return the acknowledge (following) to
Local node during an operation on Evaluation
session (Creation or Deletion):
Preceeded by : ’failure status’
Followed by : ”’ACK to return’, ’Session’.”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS encoutered a problem trying to send acknowledge of the given”
” operation to the caller. The problem could be described in a preceeding message.”
” This message could explain a DBS_COMMUNICATION_PROBLEM or COMMS_TIME_OUT ”
” status given on a local node.”
RECOVERY_END:

DBS_ERR_505 : The operation ARCHIVE_EVALUATION_SESSION failed because the
Session state is not set correctly. The Local DBS should have
set it to the value ’To Be Archived’. Then it is an Internal Problem:
Followed by : ”’Session Name’”.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS found an invalid value for an Evaluation Session State.”
” It means that Local Nodes called a particular combination of operations”
” (e.g. ARCHIVE and DELETE) at the same time and that they both reset”
” the Execution Session State without detecting the ”
” other request. This should not lead to any inconsistency. ”
” Note other messages related to the same session. ”
RECOVERY_END:

DBS_ERR_506 : The operation RETRIEVE_EVAL_SESSION_FROM_FA failed when
Central DBS tried to update the Oracle references of the files
succesfully copied from the Final Archive device. The Reference
updates were interrupted by this problem. Files and Session
references are not updated.
This Problem can be recovered by the recovery scripts:
Followed by : ”’Session Name’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the RETRIEVE_EVAL_SESSION_FROM_FA operation. The retrieving of files from FA ,”
” succeeded, but the update of Oracle file references failed.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_507 : The operation RETRIEVE_EVAL_SESSION_FROM_FA failed when
Central DBS tried to update the Oracle references of the Session.
The Session has been succesfully copied from the Final Archive device, but

the Reference updates were interrupted by this problem : Files and Session references are not updated.
This Problem can be recovered by the recovery scripts (same as last error):
Followed by : ”’Session Name’”.
RECOVERY_START:
” The operation RETRIEVE_EVAL_SESSION_FROM_FA failed when Central DBS tried ”
” to update the Oracle references of the Session. The Session has been succesfully”
” copied from the Final Archive device, but the Reference updates were interrupted by”
” this problem: Files and Session references are not updated.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_508 : The operation RETRIEVE_EVAL_SESSION_FROM_FA failed when Central DBS tried to update the Oracle references of the files.
The Session has been succesfully copied from the Final Archive device, but the Reference updates were interrupted by this problem : Files and Session references are not updated.
This Problem can be recovered by the recovery scripts (same as last error):
Followed by : ”’Session Name’”.
RECOVERY_START:
” The operation RETRIEVE_EVAL_SESSION_FROM_FA failed when Central DBS tried ”
” to update the Oracle references of the files. The Session has been succesfully”
” copied from the Final Archive device, but the Reference updates were interrupted by”
” this problem: Files and Session references are not updated.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_509 : The operation RETRIEVE_EVAL_SESSION_FROM_FA failed because the Session state is not set correctly. The Local DBS should have set it to the value ’To Be Imported’ as a New Name is given in arguments.
Then it is an Internal Problem:
Followed by : ”’Session Name’, ’Session State’”.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS found an invalid value for an Evaluation Session State.”
” It means that Local Nodes called a particular combination of operations”
” (e.g. RETRIEVE and DELETE) at the same time and that they both reset”
” the Execution Session State without detecting the ”
” other request. This should not lead to any inconsistency. ”
” Note other messages related to the same session. ”
RECOVERY_END:

DBS_ERR_510 : The operation IMPORT_EVAL_SESSION_FROM_FA failed because the Session state is not set correctly. The Local DBS should have set it to the value ’To Be Imported’.
Then it is an Internal Problem:

Followed by : ”’Session Name’, ’Session State’”.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS found an invalid value for an Evaluation Session State.”
” It means that Local Nodes called a particular combination of operations”
” (e.g. IMPORT and DELETE) at the same time and that they both reset”
” the Execution Session State without detecting the ”
” other request. This should not lead to any inconsistency. ”
” Note other messages related to the same session. ”
RECOVERY_END:

DBS_ERR_511 : DBS failed to remove the directory structure of an Evaluation Session during the operation DELETE_EVALUATION_SESSION.
The Oracle References are updated (commited) but some (all) Session files are still on the TRDB disk.
All the Session files on the TRDB disk must be deleted by the recovery scripts.
Followed by : ”’Status of failure’, ’Session Name’”.

RECOVERY_START:
” Central DBS failed to remove the directory structure of an Evaluation.”
” Session during the operation DELETE_EVALUATION_SESSION.”
” The Oracle References are updated (commited) but some (all)”
” Session files are still on the TRDB disk. ”
” All the Session files on the TRDB disk must be deleted by the Recovery ”
” Scripts <Delete Session> Menu. ”
RECOVERY_END:

DBS_ERR_512 : SESSION_STATE_FAILURE
During the export operation, the session state cannot be set back to previous state given as argument. This is a severe error because Session could be locked in a state where all action is forbidden.
Followed by : ”’previous session state’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the EXPORT_EVALUATION_SESSION operation. The session state cannot be set back”
” to previous state given as argument.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_513 : MOVE_SESSION_DIR_SPACE_FAILURE
Retrieve of an Evaluation session: the File structure has been copied from the Final Archive disk into the TRDB_DISK, in the SESSION_NAME root directory.
Then, this file structure has to be renamed into the NEW_NAME root directory as requested by the User. The renaming failed, leaving the TRDB in an inconsistent state : – Session NEW_NAME is set to the state TO_BE_IMPORTED (SESSION_IS_USED)
– Data retrieved is kept in the SESSION_NAME root directory
Followed by : ”’SESSION_NAME’, ’NEW_NAME’, ’STATUS of failure’”.
RECOVERY_START:

” Central DBS encoutered a problem during the execution of”
” the retrieving of an evaluation session. ”
” The file structure has been copied from the Final Archive disk into the TRDB_DISK, ”
” in the SESSION_NAME root directory.”
” This file structure has to be renamed into the NEW_NAME root directory as”
” requested by the User. The renaming failed, leaving the TRDB in an inconsistent state:”
” – Session NEW_NAME is set to the state TO_BE_IMPORTED (SESSION_IS_USED),”
” – Data retrieved is kept in the SESSION_NAME root directory.”
” Use the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_514 : EVAL_DIR_SPACE_COMPLETION_FAILURE
Retrieve of an Evaluation session: the File structure has been copied from
the Final Archive disk into the TRDB_DISK, in the NEW_NAME root directory.
Then, this file structure has to be completed to allow additionnal storage
of data (i.e. an evaluation session retrieved with a New Name creates an open
session where data can be added). The file structure completion failed leaving
the TRDB in an inconsistent state :
– Session NEW_NAME is set to the state TO_BE_IMPORTED (SESSION_IS_USED)
– NEW_NAME file structure is not complete.
Followed by : ”’SESSION_NAME’, ’NEW_NAME’, ’STATUS of failure’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the retrieving of an evaluation session. ”
” The file structure has been copied from the Final Archive disk into the TRDB_DISK, ”
” in the NEW_NAME root directory.”
” This file structure has to be completed to allow additionnal storage of data (i.e. an evaluation”
” session retrieved with a New Name creates an open session where data can be added).”
” The file structure completion failed leaving the TRDB in an inconsistent state:
” – Session NEW_NAME is set to the state TO_BE_IMPORTED (SESSION_IS_USED),”
” – NEW_NAME file structure is not complete.”
” Use the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_801 : COMMS INIT FAILURE.
During a FA controller start, communication initialisation failed.
The FA controller is stopped.
Followed by : ”’channel number’ returned ’status’”.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
”No Recovery Script action has to be executed following that message.”
” DBS Central Archive has not been able to start due to Communication”
” Services problems. Refer to the status given in the message.”
RECOVERY_END:

DBS_ERR_802 : ACK NOT SUPPLIED.
During a dispatcher operation, a supply acknowledge operation failed.
The caller has not received the expected operation status.
The COMMS return an error code.
Followed by : ”(’status’ for ’operation’ to ’caller’)”.

RECOVERY_START:
”   This error message alone does not describe an inconsistency in the TRDB.”
”No Recovery Script action has to be executed following that message.”
”   DBS Central Archive has not been able to supply an acknowledge to the caller ;”
” (local application) ”
”   The Communication Services returned an error (part of the message).”
”   This message could explain a DBS_COMMUNICATION_PROBLEM or COMMS_TIME_OUT ”
”   status obtained on a local application.”
RECOVERY_END:

DBS_ERR_803 : INVALID COMMAND.
During a dispatcher operation, an execution operation can’t recognize a command.
Invalid command is rejected.
Followed by : ”(’command’ with ’command/controller info’)”.
RECOVERY_START:
”   This error message alone does not describe an inconsistency in the TRDB.”
”No Recovery Script action has to be executed following that message.”
”   DBS Central Archive has not reckognised a command for the Final Archive SAS.”
”   This invalid command is rejected. Analyse the command: it can be invalid”
”   due to the context (e.g. Auto Arch command without reservation of a device)”
”   or to the command syntax (included in the message).”
RECOVERY_END:

DBS_ERR_804 : INVALID DRIVE STATE.
During a job execution, a deallocate operation failed.
A drive may be locked.
Followed by : –/–
RECOVERY_START:
”   This error message alone does not describe an inconsistency in the TRDB.”
”No Recovery Script action has to be executed following that message.”
”   DBS Central Archive failed to DEALLOCATE a Final Archive drive.”
”   (e.g. after an Export). A drive may be locked following that problem”
”   Activities related to Final Archive should end as soon as possible.”
”   Stopping/restarting the Final Archive SAS will close the problem”
RECOVERY_END:

DBS_ERR_805 :   INVALID JOB STATE.
During a dispatcher operation, an update or remove operation failed.
Job has a bad state (PENDING, ALLOCATED, ACTIVE, ’removed’).
Followed by : ”(’BAD status’ for ’EXPECTED status’)”.
RECOVERY_START:
”   This error message alone does not describe an inconsistency in the TRDB.”
”No Recovery Script action has to be executed following that message.”
”   DBS Central Archive failed to remove a job for the Final Archive”
”   process (e.g. after it is executed by the FA SAS). It remains with”
”   a bad state (PENDING, ALLOCATED or ACTIVE instead of being removed).
”   This can lead to future problems as this job could be considered for”
”   a next FA SAS request. Activities related to Final Archive should”
”   end as soon as possible.”

” Stopping/restarting the Final Archive SAS will close the problem”
RECOVERY_END:

DBS_ERR_806 : STOP_FA_SAS_REQUEST_FAILURE.
Central Archive failed to send stop request to Final Archive.
Followed by : ”’channel number’ returned ’status’”.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” DBS Central Archive failed to send the stop request to the Final Archive”
” process. If no Final Archive SAS was running, this message is normal.”
” If it was running, it could show a problem in the definition of the”
” System Topology Table.”
RECOVERY_END:

D-3.4 Model Execution

D-3.4.1 HLCL on–line

When an error occurs in a command sequence during execution, the effect depends on the predefined TRAP variable. If *error trapping* is on, an error within a command interrupts the command sequence, as if the interrupt key combination had been pressed. Execution can then be resumed with the RESUME command. If the TRAP variable is set to false errors in the command sequence do not interrupt the execution of the sequence.

If *error trapping* is on the erroneous command is displayed in the ICP window and an explanatory text is displayed which indicates the type of error.

If HLCL is used as an interactive command language there are two types of errors which can occur. Figure 16 shows the way syntax errors are displayed while the user types a command on–line.

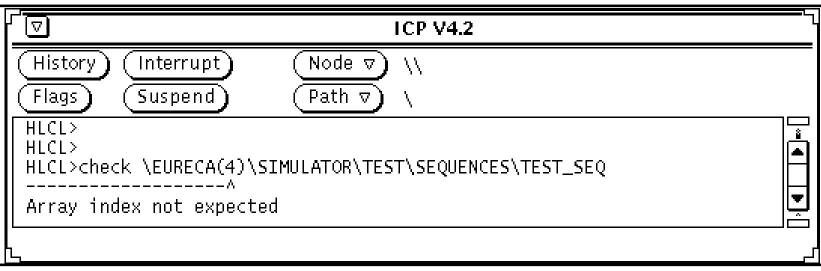


Figure 16 : *Syntax errors in the HLCL command are shown by a little arrow.*

Errors during command execution are displayed as shown in Figure 17. The error messages are written in plain and detailed text. The errors messages are self–explanatory, therefore a list of possible syntax errors is not provided.

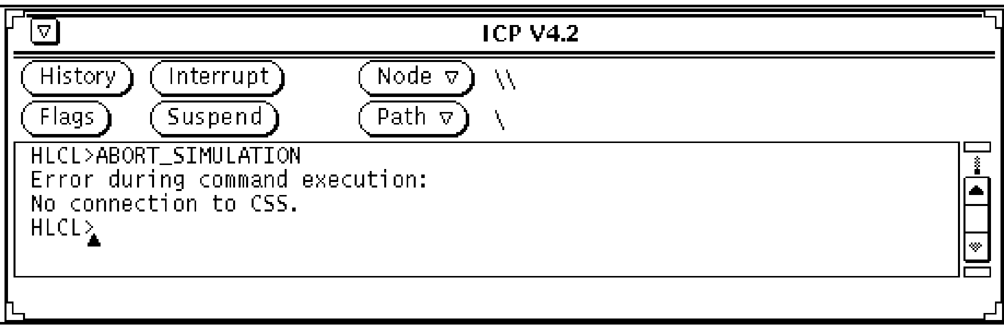


Figure 17 : *Errors during command execution*

D-3.4.2 Model Execution Messages

Errors produced during model execution are delivered to the central CGSI error services.
Refer to chapter 4.3 for more information about the CGSI error handling.

CSS Startup

Frame duration: 0.200 seconds”);

CSS Kernel started
CSS Version: CGS/CSS Version x.x from <date>”);

Buffer overflow: too many CSS events to be stored in the Test Result DataBase
(TRDB), 10 events dropped, 0 events stored
Detecting Unit: LOGGER_SPECIFIC_USERS_EVENT, Faulty Unit: CSS_EVT_HANDLER”);

CMAS software successfully started
Detecting Unit: CTU, Faulty Unit: CMAS

Could not launch CSS event handler

CSS event handler started

CSS_EVT_HANDLER: Initialization completed

CSS_EVT_HANDLER: could not get associated sessionname, use default session instead
Status returned: SESSION_NOT_FOUND

CSS_EVT_HANDLER: shutting down...

CSS part of execution session
Name of session: <name>

CSS Kernel shutting down

CSS Logger_BE: Archive messages read: 2

CSS Logger_BE: Creating archive file
File: <unix_pathname>

CSS Logger_BE: Creating data set
File: <unix_pathname>

CSS Logger_BE: Creating data set statistics file”,
File: <unix_pathname>

CSS Logger_BE: Creating logfile
File: <unix_pathname>

CSS Logger_BE: finished

CSS Logger_BE started

New Statevector loaded
Name of Statevector: <unix_pathname>

D-3.4.2.1 Error messages produced by DB Browser, MOCS and ICP

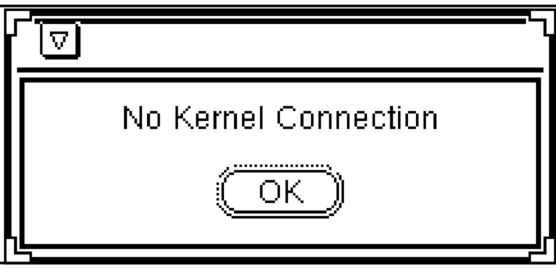


Figure 18 : *MOCS message window*

There is a special feature used by the MOCS/ICP error handling procedure. Each involved component adds one part of the error string, the result a an error string of unpredictable length. It is not possible to include all possible because the number of combinations is unlimited. An example of the resulting error string is given in Figure 19.

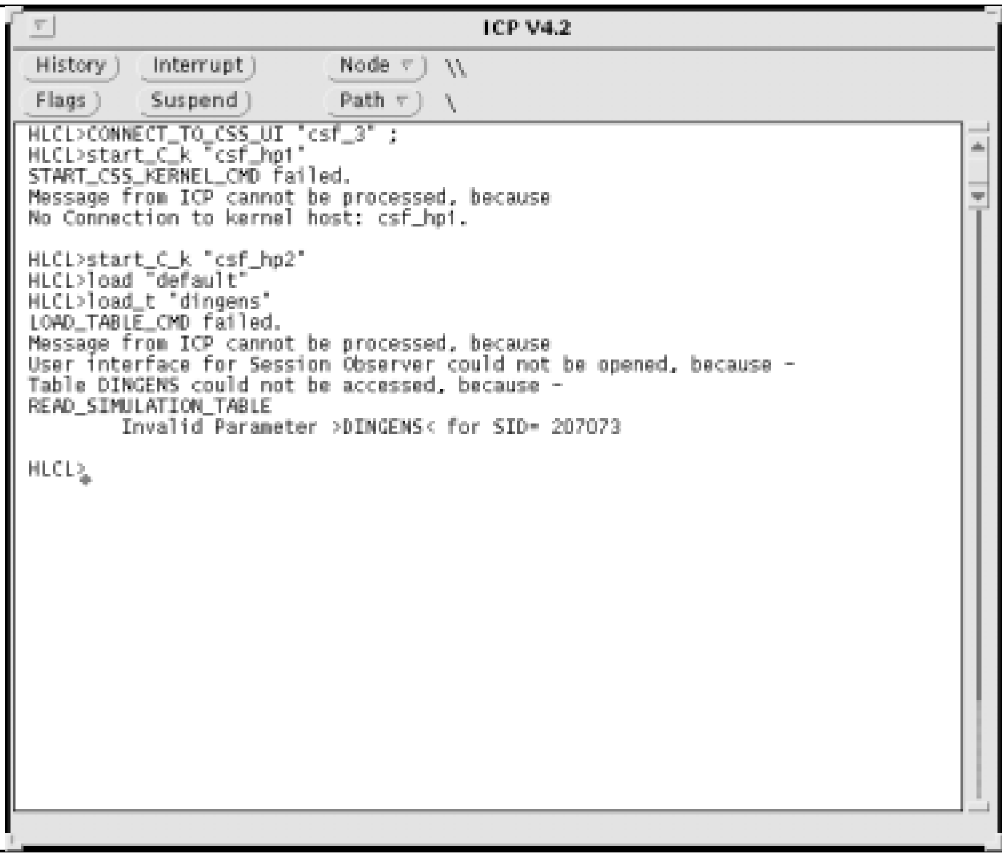


Figure 19 : *A concatenated error string is displayed in the ICP window*

D-3.4.2.2 DB Browser Messages

C

Can not access model in database.

Contact your system administrator.

D

Database crashed.

Contact your system administrator.

DB server does not respond. Try again.

Make it so and contact your system administrator if it fails permanently.

E

Environment variable TWO_TASK is not set.

It is not possible to connect to the database. CSS was not installed properly.
Contact your system administrator.

F

First select a library in side the database browser.

User error. Select a library first.

I

In order to load the model under the selected user environment, all invalid function references have to be updated in the Database. Currently the model is locked against editing by another session.

The model is in use. Try again later.

In order to load the model under the selected user environment, all invalid function references have to be updated in the Database. You have not got the necessary edit permissions.

It is impossible to open the model with the currently selected CCU. Select a CCU which fits to the model references.

M

Model creation failed.

Try again, look for further error messages.

Model removal failed.

Try again, look for further error messages.

N

<name> has been deleted or renamed.

A wrong model name is entered in a dialogue window. Type a correct i.e. existing models name.

O

Operation aborted: Model has been deleted or renamed.

There is no model with the given name. Check the spelling.

Operation aborted: Selected library is locked.

Another user is working with the selected library. Try again later.

Operation aborted: Selected model is locked.

Another user is working with the selected model. Try again later.

Operation aborted: There are still background jobs active.

Try again later.

P

Paste failed.

Try again, look for further error messages.

Paste failed: Copied model possibly has been deleted or renamed.

There is no model with the given name. Check the spelling.

Please close all open editors on the buffered model first.

Make it so.

Please close all open windows first.

Close the window(s), then repeat the command.

Q

Quit aborted: Please close all open windows first.

An attempt is made to quit the DBB while there is at least one open window. Close the window(s).

Quit aborted: There are still background processes active.

Try again later.

R

Rename failed: Model possibly has been deleted or renamed.

There is no model with the given name. Check the spelling.

S

Simulation table removal failed.

Try again.

State vector removal failed.

Try again.

T

Table creation failed.

Try again, look for further error messages.

There is nothing to paste.

First copy the object, then paste.

There is no table to paste.

First copy the table, then paste.

U

Updated inconsistent model list.

The selected model does not exist any more, it is not possible to perform the desired operation.

The CCU you selected does not fit to the model. Check your selection.

Updating of invalid function references would exceed the maximum levels of decomposition.

The currently selected CCU doesn’t fit to the model references. Choose another CCU.

D-3.4.2.3 MDE Error Messages – created during model editing

A

<u>A record has to consist of at least <number> components.</u>

Increase the record size accordingly.

<u>AIL code contains the illegal keyword <name>.</u>

Remove the illegal keyword.

<u>AIL–reserved words must not be used as names.</u>

Rename the item accordingly.

<u>All engineering units must comply with AIL syntax.</u>

Check the units (spelling).

<u>All names must comply with AIL syntax.</u>

Naming conventions are

* the first character must be a letter
* any subsequent characters must be letters, digits, or the underscore ('_')
* two underscores cannot occur together, nor can a model name end with an underscore
* a single space between letters and/or digits will be replaced by an underscore

<u>All names must comply with AIL identifier syntax.</u>

see above

<u>All pathnames must comply with Database pathname syntax.</u>

The Database pathname syntax is as follows:

* the first sign must be a backslash ('\')
* node names are separated by a backslash
* a node names consists of letters, digits or the underscore ('_')

<u>All positions at the actual edited composite are occupied.</u>

First enlarge the function block, then try again.

<u>All positions at the function block are occupied.</u>

First enlarge the function block, then try again.

<u>Attention: Name of pasted interface item has been changed.</u>

This is not an error message. Note that you can’t have two interface items with identical names assigned to the same object.

B

<u>Both value receivers are already connected to different producers. Disconnect one and try again.</u>

Make it so.

C

<u>Can not access desktop directory <name>.</u>

There is no directory with the given name. Check the spelling.

<u>Can not access Interleaf configuration files.</u>

Contact your system administration. Error caused by defective installation.

<u>Can not access model in database.</u>

There is no model with the given name. Check the spelling.

<u>Can not apply <vector or matrix> of still undefined type.</u>

First define the data type, then define the vector or matrix components.

<u>Can not find a position for a new interface item.</u>

Enlarge the function block.

<u>Can’t connect a MEASUREMENT to a MODEL INPUT.</u>

You selected a wrong onboard item type – try stimulus.

<u>Can’t connect a STIMULUS to a MODEL OUTPUT.</u>

You selected a wrong onboard item type – try measurement.

<u>Can’t connect incompatible interface items.</u>

The type definitions of input and output do not fit together. Check the type definitions.

Can’t edit interface item that is connected to an onboard item.

First disconnect the input/output, then edit it.

Compilation error.

Refer to section D-2.2.1.3 for detailed information.

Connect one of theses value receivers to a value producer first. Then try again.

Make it so.

Choose ”edit” to edit this row.

Make it so.

Choose ”toggle” to edit this row.

Make it so.

Column <number> has different action part.

It is not possible to minimize the matrix in the decision table because the column has a different action part. Select another column or remove the action part.

Copy/paste references from a non–frozen library is restricted to models in the same library.

That’s the way it is.

Connection to model interface: <name> is not onboard compatible.

Check the data type for inconsistencies.

CSS–reserved words must not be used as names.

Change the name.

Currently the model is locked for editing.

The model is in use, wait e few minutes then try again.

Currently the state codes are not defined.

First define the state codes, then try again.

D

Decision table contains the illegal keyword <name>.

Remove the illegal keyword.

Desktop directory not specified.

Contact your system administrator. Installation fault.

Directory <name> does not exist.

Check your input (spelling).

Don’t connect value producers. (Please refer to the MDE–GL Manual.)

It is not possible to connect two outputs. In any case, refer to the CGS User Manual, (chapter: Model Development)

E

Error: Initial value is out of range.

Ignore, ranges have no meaning in the current implementation.

Error: <name> does not exist.

The model name does not exist. try again with a valid name.

Error: <name> exceeds the limit of <number> characters.

Rename the object.

Error: <name> identifies a directory.

You typed a directory name instead a model name. Correct the input.

Error: No read permission for <name>.

Copy the model, then read it or ask the model owner for giving you the read permission.

Error: Selected Interface item is connected.

First cut the connection, then delete the interface item.

Error: The state code size is restricted to <number> characters.

Shorten your state code.

Error: Value is out of range.

Correct the value.

Errors in configuration.

Read the following error report for more information.

Execute create grouping link inside the graphic subview.

Move the mouse pointer into the graphic subview and repeat the command.

Execute create input inside the graphic subview.

Move the mouse pointer into the graphic subview and repeat the command.

Execute create output inside the graphic subview.

Move the mouse pointer into the graphic subview and repeat the command.

Execute paste inside the graphic subview.

Move the mouse pointer into the graphic subview and repeat the command.

External referencing is restricted to functions in frozen libraries.

You can’t make a reference to the desired library because it it not frozen.
Freeze it if possible or copy the desired function with the **copy–>normal** command.

F

Fatal Implementation Error !!!

This error should not occur.

File named <name> is an existing directory.

Wrong input. The name already exists.

First select the desired composite function block inside the hierarchy browser. Then try again.

Don‘t care about this message. It appears as soon as the hierarchy window is opened.

I

I/F items of record type may not become attached to the model interface.

The record data type may be used only within the model.

Impossible to reduce size (marked object or one of its connections outside desired area.

Move your function blocks until they fit into the desired size.

It is not allowed to edit a referenced subtree.

The language elements you try to edit are copied by reference.
Use the **copy–>normal** command instead, then edit the items.

L

Local variables contain the illegal AIL keyword <name> .

Remove the AIL keyword from the local variables definition.

M

Macro definitions contain illegal AIL keyword <name> .

Remove the AIL keyword from the macro definition.

Model internal references are not allowed.

That’s the way it is. Use the **copy–>normal** command.

Model is currently locked for editing.

It is not possible to perform the last initiated action. Try again later.

Model is locked.

It is not possible to perform the last initiated action. Try again later.

N

<name> already identifies another component.

Choose a different name.

<name> already identifies another interface item of parent function.

Choose a different name.

<name> contains unconnected reference points and can not be copied by reference.

Either connect the unconnected reference points and then try again or choose the **copy–>normal** command.

<name> does not conform with AIL syntax.

Check your input. Note that AIL syntax is Ada syntax.

<name> is locked against inspecting.

If this error occurs report it via SPR (Software Problem Report) immediately to DASA/RI.

<name> is locked against editing.

The model is in use. Try later.

<name> selected interface items cannot be cut. Maybe they are connected over hierarchy level borders.

Check your implementation and cut low level connections first.

No connection line selected. Operation aborted.

You didn’t move the mouse pointer to a connection line and click where to split the line. Try again.

No connection possible in this position. Move objects and try again.

Make it so.

No functions selected.

First select a function.

No write permission in directory <name>.

Try again with your own directory.

Not all names could be changed.

The **unify** command could not change all names. Check the implementation.

O

Only I/F items of onboard type may become attached to the model interface.

The selected input/output has a data type which can be used only within the model.

Only no matter fields may be expanded.

Select the ’no matter’ field to expand the decision table. If there is no ’no matter’ field the table has the maximum size.

Operation aborted.

Try it again.

Operation aborted – would result in connection cycle.

It is not allowed to implement asynchronous chains, break the chain.

Operation aborted: Choose a new name to break the signal.

Your input is the name of the selected signal, type a new name to break the signal.

Operation aborted: Copied functions contain references into libraries not included in the model's user environment.

The selected CCU does not contain the referenced library or libraries. Go back and select an appropriate CCU.

Operation aborted: Initial value <number> would be out of range for engineering unit <name>.

Check initial value and engineering value – doesn't fit together.

Operation aborted: No CTG host selected.

Select a host in the pop–up menu.

Operation aborted: Pasting this interface item of RECORD type would lead to name conflicts.

Check the contents of the record for name conflicts.

Operation aborted: Record structure would lead to naming conflicts.

Check the record structure for duplicate names.

Operation aborted: The AIL code contains the following lines that exceed the limit of <number> characters.

Split the AIL statements into several lines.

Operation aborted: The combination of selected functions and selected filters will not produce any output.

Correct the filter setting and/or the function selection.

Operation aborted: The composite function contains unconnected reference points and can not copied by reference.

Choose the **copy–>normal** command or correct the function block you want to reference.

Operation aborted: The output device is undefined.

Select either a TPS document or a printer as output device.

Operation aborted: No printer server selected.

Select the printer server (at least **default**). If there is no entry in the pop–up menu you found an installation error.

Operation aborted: would result in too many levels of decomposition.

The maximum levels of decomposition is 16.

P

Paste buffer contains function blocks.

It is not possible to paste a function block into an interface editor window.

Paste buffer is empty.

First copy something then paste.

Please close the MOCS window on <name> first.

Make it so.

Please close all open editors on <name> first.

Make it so.

Please select one or more Function Blocks for this operation and try again.

Make it so.

Position of last cuttet I/F item is already occupied.

Increase the size of the function block.

R

Referencing copied functions in this composite would result in loop.

Check your implementation for asynchronous loops.

Rename the top level function inside the Database Browser.

Move the mouse pointer into the DBB window and repeat the command.

S

Select exactly one object for this operation.

Make it so.

Select exactly two objects for this operation.

Make it so.

Selected interface item is already connected to selected onboard item

No error – the desired connection is already established.

Selected item does not represent a Columbus onboard item.

Check the item definition, may be it is undefined.

Selected onboard item is incompatible to selected model interface item.

Check the item definition and the model input/output definition, maybe there is an undefined item.

Sorry, this function is not implemented yet.

What a pity !

Sorry, to a top level function are not supported..

What a shame !

Spaces have been replaced with ‘_’.

Ada syntax does not allow names with spaces inside.

T

The frame has to be an integer value between <min–number> and <max–number>.

Select the frame within the given limits.

The global symbol is connected to onboard. Cannot accept non onboard types.

Connect only to onboard types.

The interface of <name> still contains grouping links. You have to remove all of them prior to converting a composite to an atomic function block.

Do not use the convert function. Create a new function block if you need one.

The last executed command cannot become undone.

Try it manually.

The limit of <number> record components is already reached.

Size the record within the limits.

The maximum number of actions is limited to <number>.

Reduce the actions in the decision table to the given number.

The maximum number of conditions is limited to <number>.

Reduce the conditions in the decision table to the given number.

The minimum matrix size is <number> rows and <number> columns.

Increase the number of elements.

The minimum number of record components is <number>.

Increase the number of elements.

The minimum vector size is <number>.

Increase the number of elements.

The model has not been modified since last saving.

The **save** operation will not be performed.

The model’s user environment does not contain the selected library of onboard items.

Go back and select the appropriate CCU (with the correct onboard items).

The name was too long and has been truncated to the legal length of 16 characters.

If you don‘t like the truncated name, delete the object and create a new one.

The number of elements is limited to <number>.

Reduce the number of elements.

**The objects to paste don‘t fit with the size of the Composite Function Block.
First choose a larger size then try again.**

Make it so.

The referenced function is located in a library not visible in the current user environment.

The reference can’t be resolved. Select a CCU which contains the referenced function library.

**The rotated object won’t fit with the size of the Composite Function Block.
First choose a larger size then try again.**

Make it so.

The selected definition point is unconnected.

It is not possible to show any references because the definition point is unconnected.

The selected interface item is connected to the interface of a reference function. You are not allowed to modify reference functions.

Replace the reference function by a copy of that function. Then try again.

The selected reference point is unconnected.

It is not possible to show any references because the reference point is unconnected.

The selected signal entry is unconnected.

It is not possible to disconnect this signal because it’s already unconnected.

The selected value receiver is connected to another producer. Disconnect it before making another connection.

It is not allowed to connect an input with two different outputs.

The type has to be defined in order to set the initial value.

First define the type, then set the initial value.

The type of a selected block could not be changed.

It is not possible to change the type. Delete the block and create a new one with the desired type.

There are no objects to paste.

First copy or cut an item, then use the paste command.

There are no type inconsistencies in the selected global symbol.

Enjoy, everything is correct.

There is already an open AIL Editor.

You can’t get two editor window on the same object.

There is already an open Composite Editor.

You can’t get two editor window on the same object.

There is already an open Composite I/F Editor.

You can’t get two editor window on the same object.

There is already an open Documentation Tool.

You can’t get two documentation tools on the same object.

There is already an open Hierarchy Browser.

You can’t get two browser on the same object.

There is already an open Icon Editor.

Note that there is already an open editor window, close one.

There is no appropriate parent object for the new interface item.

This means ’All positions at the function block are occupied.’ Enlarge the function block and try again.

There is no column with an appropriate constellation of conditions.

It is not possible to minimize the matrix in the decision table. Select another condition field.

There is no icon to paste.

First **copy** the icon, then use the **paste** command.

There is no onboard reference specified.

Select an onboard reference first.

There is no type compatible Definition Point.

You can only connect Reference and Definition points of the same type.

There is no unconnected Signal.

Disconnect a signal before you try to re–connect it.

These object are already connected.

Disconnect other connections, then try again.

This code generation process is still busy; the model is locked against saving.

Code generation can take several hours, watch the process and repeat your command as soon as the code generation is finished.

THIS ERROR SHOULD BE IMPOSSIBLE: <name>.

If this error occurs report it immediately (write a Software Problem Report).

This feature is currently not implemented.

What a pity !

This global symbol is connected to interface item(s) of reference function(s). The set of types had to be restricted to the type of the interface item(s) of the reference function(s).

Replace the reference function by a copy of the function. Then try again.

This global symbol is connected to multiple type incompatible interface items of reference functions. You are not allowed to modify reference functions.

Check if you use the referenced function block correctly.

This global symbol is referenced in a referenced subtree and can not be renamed.

You can delete the subtree and choose the **copy–>normal** function to copy the subtree. After that a rename is possible (but not recommended).

This interface item can only be attached to composite function blocks.

Make it so.

This logical grouping does not contain any signals.

First define signals, then connect them.

This model is still busy and locked against editing.

Wait a few minutes, then repeat the last command.

This model is still busy and locked against printing.

Wait a few minutes, then repeat the last command.

This model is still busy and locked against saving.

Wait a few minutes, then repeat the last command.

This name already identifies another object. Choose a different name and try again.

Make it so.

This signal is referenced in a referenced subtree and can not be renamed.

You can delete the subtree and choose the **copy–>normal** function to copy the subtree. After that a rename is possible (but not recommended).

Y

You can't add interface items to global symbols. Please refer to the MDE–GL Manual.

Reference points and definitions points have exactly one input/output. You can't add another input/output.
In any case, refer to the CGS User Manual (chapter: Model Development).

You can't add interface items to parameter blocks. Please refer to the MDE–GL Manual.

Parameter blocks exactly one output. You can't add output.
In any case, refer to the CGS User Manual (chapter: Model Development).

You have to select an onboard item inside the Database Browser.

Press the DMS button in the DBB to switch to the onboard item selection.

D-3.4.2.4 Compilation errors from the CSS runtime system (CTG)

CTG is still in the development stage so changes are expected.

A

ADALIB_CREATION_ERROR: cannot create Ada library <name>; see error file <name>.

Please read the error file which is located in your $HOME/????. In this file you find a detailed error description.

C

CTG_CALL_ERROR: Wrong number of arguments; Usage: css_ctg_server <port_file_name>.

Please write a Software Problem Report (SPR) immediately.

D

DIRECTORY_CREATION_ERROR: cannot create directory <name>.

Please write a Software Problem Report (SPR) immediately.

E

ENVIRONMENT_ERROR: The following environment variable is not set: <name>.

Please contact your System Administrator. CSS was not installed correctly.

F

FILE_ACCESS_ERROR: <error_code> raised during accessing the file <name>.

Please check your installation for quota problems or missing mounts to other machine/programs.

FILE_ALREADY_EXISTS_ERROR: The file or directory <name> already exists.

Please write a Software Problem Report (SPR) immediately.

FILE_COPY_ERROR: cannot copy file or directory <name> to file resp directory <name>.

Please check your installation for quota problems.

FILE_CREATION_ERROR: cannot create file or directory <name>.

Please write a Software Problem Report (SPR) immediately.

FILE_DELETION_ERROR: cannot delete file or directory <name>.

Please write a Software Problem Report (SPR) immediately.

FILE_NOT_FOUND_ERROR: The file <name> does not exist.

Please write a Software Problem Report (SPR) immediately.

FILE_RENAMING_ERROR: cannot move file or directory <name> to file resp directory <name>.

Please write a Software Problem Report (SPR) immediately.

I

INCONSISTENT_SYSTEM_ERROR: the client and the server are incompatible: version_nr. of client: <number>; version_nr of server: <number>.

Please contact your System Administrator. CSS was not installed correctly.
Please write a Software Problem Report (SPR) immediately.

INTERNAL_ERROR: <error_code> raised in subprogram <name>.

Please write a Software Problem Report (SPR) immediately.

INVALID_SERVICE_ERROR: provided and required requests are incompatible: provide request <name>; required requests <name>.

Please write a Software Problem Report (SPR) immediately.

N

NO_CONNECTION: cannot establish connection between server and client; or connection to client broken.

This is a configuration problem. Please check your installation for network problems or missing mounts to other machines.

P

PORT_CREATION_ERROR: cannot create new port.

This is a configuration problem. CSS requires port numbers in the range from 8000 to 8999.

D-3.4.2.5 MOCS error messages

- *Note that the % sign will be replaced by the actually used parameter.*
- *Note that the – sign denotes that the error string will be concatenated with another error string.*

A

Are you sure, that you want the scale steps to be greater than the range ?

The scale steps will not be visible. Check your input.

C

Cannot associate definition to definition table: %.

The user tried to log an item or trace a function block. He/she has to specify a Simulation table to associate the definition with.
Correct the error by either loading the specified table table first or specify a different table.

Cannot close CSS User Interface, because – %.

Check the complete error message for the possible reason.
Try the HLCL STOP_CSS_UI command.

Cannot connect to kernel, because – %.

Check the complete error message for the possible reason.

Cannot interpret input stream: % was not expected in state %

Internal error message. Please write a Software Problem Report (SPR) immediately.

Cannot load Kernel/Mapping Table from DB.

Check the database connection. If there is no problem , this is an internal error message.

Cannot process event % because – % is not defined.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Cannot read complete records.

The user referenced a CSS record type variable in HLCL. Only record elements can be referenced. Change your HLCL command accordingly.

Cannot read mapping table because –

Check the complete error message for the possible reason.
Please contact your System Administrator. It might possibly be an incorrect CSS installation.

Cannot scan % in specification, because –

Internal error message. Please write a Software Problem Report (SPR) immediately.

Cannot start kernel, because %.

Check the complete error message for the possible reason.

Command could not be executed, because – %

Check the complete error message for the possible reason.

Command Interpreter could not be started, because – %.

Check the complete error message for the possible reason.
The reason could be a communication error raised during the connection building phase, e.g. the port number is undefined or still in use.

Command was successfully submitted to simulator. You can view it by pressing SHOW COMMANDS.

This is not an error message.

Connection timeout expired for connection to %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Connection to % broken.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Could not determine inconsistent entries, because –

Check the complete error message for the possible reason.

Could not encode % in a string, because –

Check the complete error message for the possible reason.

could not write % to database, because –

Check the complete error message for the possible reason.

could not write % to file, because –

Check the complete error message for the possible reason.

CSS cannot connect to Kernel because –

Check the complete error message for the possible reason.

D

Daemon for ICP Connection could not be started, because –

Check the complete error message for the possible reason.
Perhaps because the port name used was undefined or the port number was still occupied bey another process.
If the port number is still in use (see rest of the error message), retry later.

Database Identifier (SID) could not be determined.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Definitions could not be stored, because %

Check the complete error message for the possible reason.

E

Error accessing kernel identification, because – %

Check the complete error message for the possible reason.

F

Function cannot be traced because %

Check the complete error message for the possible reason.

H

HLCL command is not allowed in this system state.

Check in the User Manual, in which context this command is allowed.

I

Identifier Mapping Table File % not found.

Check the database connection. If the connection is ok, this is an internal error message.

Illegal step count % entered. Step count must be a numeric value.

The input must be a number greater 0.

Illegal step count % entered. Step count must be positive.

The input must be a number greater 0.

Illegal % syntax.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Incompatible SW versions of MOCS and %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Invalid input, try again.

Check your input for correctness.

Item cannot be locked, because %

Check the complete error string for the specific reason.

Item named % cannot be monitored or logged.

Ensure that the item is either external (top level I/O) , atomic or an output item.

Item named % may not be assigned values.

This value does not allow assignments. Check the data type and ensure that the item is either a top level input or a model output.

K

Kernel crashed during start up, because – %

Check the complete error message for the possible reason.

<u>Kernel did not respond to command in time.</u>

Try again.

<u>Kernel did not start in time.</u>

Try again.

<u>Kernel Interface File could not be accessed, because – %</u>

Check the complete error message for the possible reason.

<u>Kernel is disconnected.</u>

A command was given which is not allowed in the current model state.

<u>Kernel Startup Warning: %</u>

Check the complete error message for the possible reason.

L

**<u>Local command action could not be performed, because – %
!!! attention: system may be inconsistent, stop and restart !!!</u>**

Check the following complete error message for a description of inconsistencies. Then stop and restart if necessary.

<u>Logfile and Archivfile can be found at the locations: % and %</u>

This is not an error message. It simply informs the user where to find the log– and archive–files.

M

<u>Message coming from % with the operation code % cannot be interpreted.</u>

Internal error message. Please write a Software Problem Report (SPR) immediately.

<u>Message could not be encoded, because –</u>

Check the complete error message for the possible reason.

Message could not be send to kernel, because – %

Check the complete error message for the possible reason.

Message from ICP cannot be processed, because %

Check the complete error message for the possible reason.

Minimum value may not be equal to maximum value.

Check your input.

Minimum value must be less than maximum value.

Check your input.

More than one Kernel for the specified model is running.

Note that the connection to a Kernel can only be established if exactly one Kernel is running on the host. Make sure that only one Kernel is running.

N

Negative virtual factor % is illegal input.

Check your input for correctness. Then try again.

No CMAS host specified.

If you want to start CMAS, you have to specify a host machine for CMAS.

No connection to kernel host: %

Check the complete error message for the possible reason.
One possible reason: a host name has been specified, but before starting the Kernel on that host, the existence of the host in the network is tested and the test failed.

No item named % could be found.

The model does not contain any item with the specified name. Check your parameter list for correctness.

No item with destination % selected.

The user tried to remove a tracing definition from an item which is not traced.

No kernel for the specified model is running on host %.

Check your command for correctness. If necessary start the simulator on the desired host and try again.

No kernel host specified.

The user pressed the kernel start button, but did not specify the host on which the kernel should be started. Try again.

No kernel is configured for %

It is not possible to start the desired model. Check whether the model is configured for the corresponding machine type. Re–configure the model if necessary.

No open connection message from % received. Received OP–code % instead.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No path for RID % found.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No path for SID % found.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No privilege to log.

The user has no log privilege. Use the HLCL command **GRANT** to grant he specified privilege or choose the screen as destination for snapshots/traces.

No RID found for SID %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No RID for path % found.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No rule is specified for state %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No SID for RID % found.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No simulation table selected.

First select a simulation table, then repeat the command.

Non–numeric virtual factor % entered.

Check your input for correctness.

P

Pathname does not contain the correct model library.

Check the pathname for correct input.

Pathname % does not exist.

Check the pathname for correct input.

Previous command is still in execution.

Wait until the command is finished, then try again.

Programming error in ItemIdentificator: Illegal item reference.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Programming error in ItemIdentificator: No item reference available.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Protocol error in message with operation code %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Pulse type variables can only be set.

It not possible to reset a pulse variable.

R

Received unexpected reply with identification: %

Internal error message. Please write a Software Problem Report (SPR) immediately.

S

Scanning error occurred at position %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Semantics error, unsuccessfully sent message %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

SID for pathname % not found.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Simulation command % is not known to the message encoder.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Snapshot of value could not be taken, because %

Check the complete error message. Probably the interface item identification was not legal.

Specified item % is already locked.

The user tried to log an item, which is already locked. Check your input for the correct item name.

Specified item % is not yet already logged.

The user tried to remove a logging definition which does not exist.

Specified item % is not yet already monitored.

The user tried to remove a monitoring definition which does not exist.

T

Table % could not be accessed, because – %

Check the complete error message for the possible reason.

The HLCL Command Interpreter send ERROR: %

This is not a MOCS error.
Report the error message to the HLCL maintenance team.

The HLCL Command Interpreter send FATAL: %

This is not a MOCS error.
Report the error message to the HLCL maintenance team.

The item % is already monitored with different monitoring parameters.

The user tried to monitor an item, which is already monitored elsewhere on the screen. Check your input for the correct item name.

The function block % is already traced to the %.

The user tried to trace a function block, which is already to the specified destination. To change the destination first remove the trace option, then try again.

The service point directory % cannot be accessed, because %

Check the complete error message for the possible reason.

The service point file % cannot be interpreted, because %

Check the complete error message for the possible reason.

The specified function block % was not traced to the %.

The user tried to remove a tracing definition, which was not traced to the specified destination. Try again with the correct destination.

The value returned on the read item command was not the value of the expected model item.

Internal error message. Please write a Software Problem Report (SPR) immediately.

This branch may not be passed: %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

This command may not be cancelled.

Only time tagged commands can be cancelled. The wait command may not be cancelled.

This path may not be passed: %

Internal error message. Please write a Software Problem Report (SPR) immediately.

Tried to store a value in a ConditionEvaluation, which is not a value holder.

Internal error message. Please write a Software Problem Report (SPR) immediately.

U

Unexpected event with operation code % received from %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Unknown character % (%) was found while scanning.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Unknown execution status % of acknowledgement.

Internal error message. Please write a Software Problem Report (SPR) immediately.

User interface for % could not be opened, because – %.

Check the complete error message for the possible reason.

V

Value type is incorrect.

Check your input for correctness.

Variables of type Pulse/Burstpulse cannot be monitored or logged.

If the user wants to ensure, that the event occurred, the event must be registered in a counter for instance. A counter must be included explicitly in the model implementation for this purpose.

Variables of type Pulse cannot be logged. Pulse type variables are excluded from the command.

Check your input.

Variables of type Pulse cannot be monitored. Pulse type variables are excluded from the command.

Check your input.

%

% is not an atomic function block.

The operation requires an atomic function block as parameter. Check your parameter list.

% is not an interface item.

The operation requires a top level input or a model output as parameter. Check your parameter list.

% must be specified.

Check the first part of the error message for the item identification.

...

... see MOCS console for kernel error messages.

This error message occurs whenever the execution of a command by the simulator was not successfully terminated. Check the MOCS console window for a detailed error message.

D-3.4.2.6 Runtime Error Messages created during model execution

A

ADA I/O Error: STATUS_ERROR.

This exception is raised by input–output operations.
The exception is raised by an attempt to operate upon a file that is not open, and by an attempt to open a file that is already open.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA I/O Error: MODE_ERROR.

This exception is raised by input–output operations.
The exception is raised by an attempt to read from, or test for the end of, a file whose current mode is OUT_FILE, and also by an attempt to write to a file whose current mode is IN_FILE.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA I/O Error: NAME_ERROR.

This exception is raised by input–output operations.
The exception is raised by a call of CREATE or OPEN if the string given for the parameter NAME does not allow the identification of an external file.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA I/O Error: USE_ERROR.

This exception is raised by input–output operations.
The exception is raised if an operation is attempted that is not possible for reasons that depend on the characteristics of the external file.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA I/O Error: DEVICE_ERROR.

This exception is raised by input–output operations.
The exception is raised if an input–output operation cannot be completed because of the malfunction of the underlying system.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.
Report this error to your responsible system administrator immediately.

ADA I/O Error: END_ERROR.

This exception is raised by input–output operations.
The exception is raised by an attempt to skip (read past) the end of a file.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA I/O Error: DATA_ERROR.

This exception is raised by input–output operations.
The exception may be raised by the procedure READ if the element cannot be interpreted as a value of the required type. This exception is also raised by a procedure GET if the input character sequence fails to satisfy the required syntax, or if the value input does not belong to the range of the required type or subtype.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA I/O Error: LAYOUT_ERROR.

This exception is raised by input–output operations.
The exception is raised in text input–output if a value returned exceeds COUNT'LAST.
The exception is also raised on output by an attempt to set column or line numbers in excess of specified maximum line or page length, respectively.
It is also raised by an attempt to PUT too many characters to a string.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA Runtime Error: CONSTRAINT_ERROR.

This exception is predefined in the Ada language.
The exception is raised in any of the following situations:
upon an attempt to violate a range constraint, an index constraint or a discriminant constraint;
upon an attempt to use a record component that does not exist for the current discriminant values;
and upon an attempt to use a selected component, an indexed component, a slice, or an attribute, of an object designated by an access value, if the object does not exits because the access value is null.
Refer to the Programming Language Ada Reference Manual for more Information, then
check your model for possible rule violations.

ADA Runtime Error: NUMERIC_ERROR.

This exception is predefined in the Ada language.
The exception is raised by the execution of a predefined numeric operation that cannot deliver a correct result (within the declared accuracy for real typed);
this includes the case where an implementation uses a predefined numeric operation for the execution, evaluation, or elaboration of some construct.
Refer to the Programming Language Ada Reference Manual for more Information, then
check your model for possible rule violations.

ADA Runtime Error: PROGRAM_ERROR.

This exception is predefined in the Ada language.
The exception is raised upon an attempt to call a subprogram, to activate a task, or to elaborate a generic instantiation, if the body of the corresponding unit has not yet been elaborated.
This exception is also raised if the end of a function is reached;
or during the execution of a selective wait that has no else part, if this execution determines that all alternatives are closed.
Finally, this exception may be raised upon an attempt to execute an action that is erroneous, and for incorrect

order dependences.
Refer to the Programming Language Ada Reference Manual for more Information, then check your model for possible rule violations.

ADA Runtime Error: STORAGE_ERROR.

This exception is predefined in the Ada language.
The exception is raised in any of the following situations:
when the dynamic storage allocated to a task is exceeded;
during the evaluation of an allocator, if the space available for the collection of allocated objects is exhausted;
or during the elaboration of a declarative item, or during the execution of a subprogram call, if storage is not sufficient.
Refer to the Programming Language Ada Reference Manual for more Information, then check your model for possible rule violations.

ADA Runtime Error: TASKING_ERROR.

This exception is predefined in the Ada language.
This exception is raised when exceptions arise during intertask communication.
Refer to the Programming Language Ada Reference Manual for more Information, then check your model for possible rule violations.

ADA Runtime Error: OTHER_ERROR.

Any other error which doesn't fit into the categories the the predefined runtime errors will be reported here.

ADA: Case selector undefined.

This is an internal ICD problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

C

Constraint Exception found in CTU.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

CTU Exception raised during rendezvous in Timeout–Entry.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

CTU Exception raised during reception of ZERO_LOAD_INDICATOR–Response.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

CTU Fatal–ERROR: Exception raised during initialization phase.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

CTU raised an exception while in ’selective accept’.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

E

Error allocating new memory.

This is a memory overflow problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

Error during initialization: Shared–memory–Error.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

Error during initialization: Missing acknowledge from Adaptation–System.

Check your configuration i.e. for missing executables.

F

Failure during CTU’s initialization phase.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

N

Numeric Exception found in CTU.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

P

Program Exception found in CTU.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

S

Storage Exception found in CTU.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

T

Tasking Exception found in CTU.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

O

Other Exception found in CTU.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

R

Resuming monitoring.

This is not an error message – the monitoring was suspended because there are too many monitoring values. Now it’s starting again.

T

Too many monitoring values – suspend monitoring.

This is an internal traffic problem. Too many monitoring items are send which cannot be handled by the receiver.
Change the simulation table by reducing the number of monitoring items and try again.

U

Unable to close logfile.

You have no permission to write a log file. Check the contents of the environment variable CSS_LOG_DIR and change it accordingly or
Check the available free storage space on your account.

Unable to open archive file.

You have no permission to open an archive file. Check the contents of the environment variable CSS_LOG_DIR and change it accordingly or
Check the available free storage space on your account.

Unable to open logfile.

You have no permission to open an logfile. Check the contents of the environment variable CSS_LOG_DIR and change it accordingly or
Check the available free storage space on your account.

D-4 Final Archiving

DEV_CTRL_PROBLEM raised in FA_SAS_MUX.RUN_MUX

E_FA_SAS. Cause: Fa sas error. Device Controller Init failed with FA_UNIX_PROBLEM; Final Archive Init Aborted.. Consequence: Current operation fails. Recovery act: None. Debug info: None.

Unix error : ENOENT in DISC_ACCESS.INITIALIZE.

E_FA_SAS. Cause: Fa sas error. GET_FILE_SIZE(device_file) with /dev/dsk/c1t3d0s2 returns : No such file or directory. Consequence: Current operation fails. Recovery act: None. Debug info: None.

D-5 Test Evaluation

D-5.1 TEV Error Report Basics

The minimum information given to the user is a context–driven message displayed in the frame footer of the concerned window after each user transaction. If an error occurs then error status will be added.

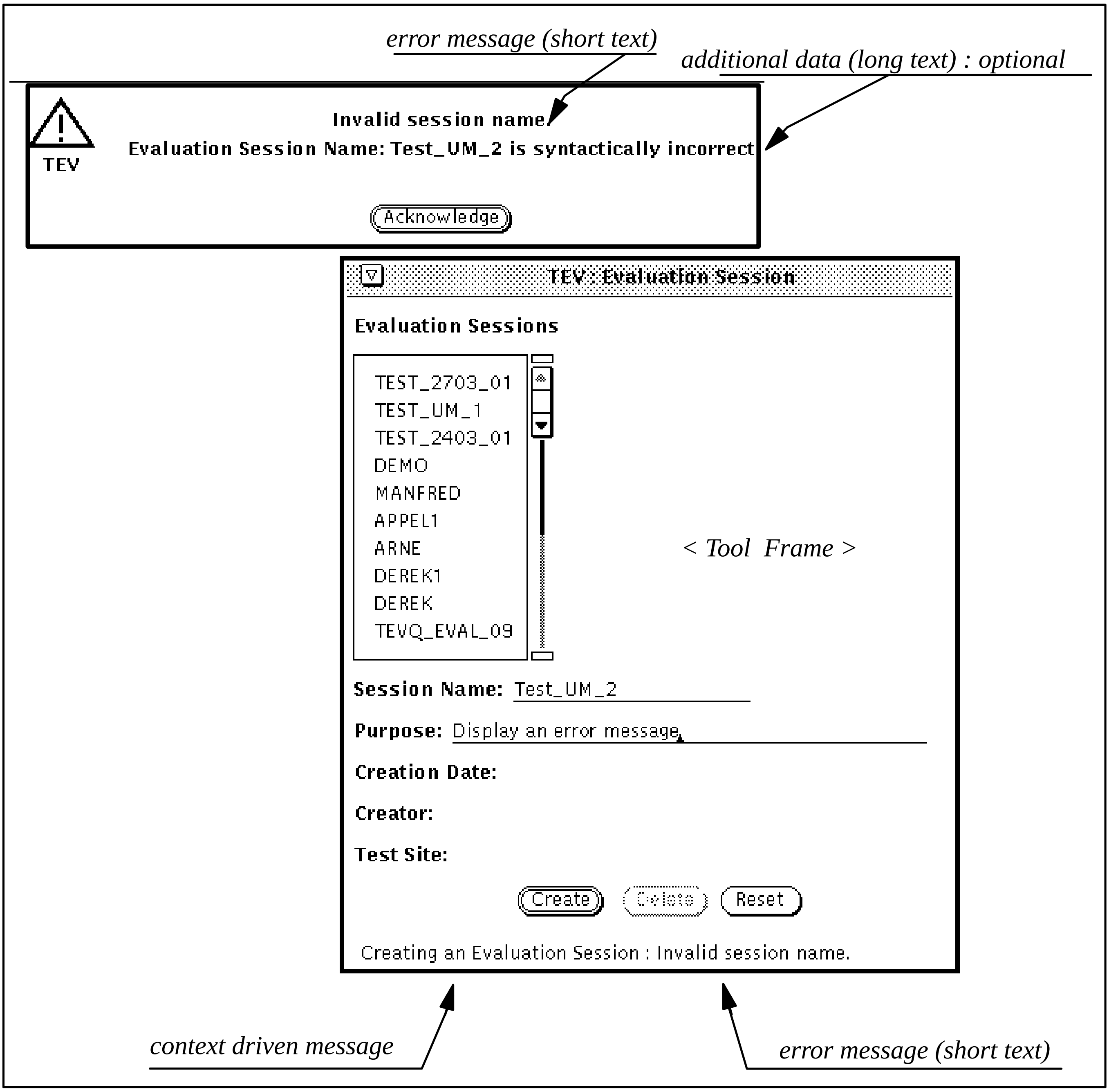


Figure 20 : *Error Report example in TEV*

D-5.1.1 DBS error messages displayed in TEV

Note that different DBS errors may be displayed with a unique TEV error message text.

TEV reports the following errors for each of the DBS errror status :

DBS_ERROR	**TEV_ERROR**
INDEX_ERROR	TEV_INTERNAL_ERROR
LIST_NOT_NULL	TEV_INTERNAL_ERROR
SESSION_NOT_FOUND	SESSION_NOT_FOUND
INVALID_SESSION_NAME	INVALID_SESSION_NAME
EXISTING_SESSION	SESSION_ALREADY_EXISTING
SESSION_IS_USED	SESSION_IS_USED

A SESSION_IS_USED error will be reported if a user tries to delete, archive or export a session which is not exclusively allocated to him.

SESSION_IS_ALLOCATED	SESSION_IS_ALLOCATED
SESSION_IS_ARCHIVED	SESSION_IS_ARCHIVED
SESSION_NOT_ARCHIVED	SESSION_NOT_ARCHIVED
SESSION_IS_EMPTY	SESSION_IS_EMPTY
SESSION_IS_ONLINE	SESSION_IS_ONLINE
RETRIEVING_NOT_AUTHORISED	SESSION_IS_ONLINE
DATA_ALREADY_ONLINE	DATA_ALREADY_ONLINE
DATA_PARTIALLY_ONLINE	DATA_PARTIALLY_ONLINE
DATA_NOT_ONLINE	DATA_NOT_ONLINE
EXISTING_FILE	FILE_ALREADY_EXISTING
FILE_NOT_FOUND	FILE_NOT_EXISTING
DATA_NOT_FOUND	DATA_NOT_FOUND
EXISTING_USER	EXISTING_USER
PRINTING_REROUTED	PRINTING_REROUTED

DBS_ERROR	**TEV_ERROR**
FILE_IS_EMPTY	FILE_IS_EMPTY
CANCEL_NOT_AUTHORISED	CANCEL_NOT_AUTHORISED
JOB_CANCELLED	JOB_CANCELLED
FA_MEDIUM_NOT_AVAILABLE	FA_ERROR
FA_MMI_REQUEST_CANCELLED	FA_ERROR
FA_RESERVED_FOR_AUTO_ARCH	FA_ERROR
FA_DEVICE_NOT_ALLOCATED	FA_ERROR
ACTIVE_FA_JOB LIST_NULL	FA_ERROR
INSUFFICIENT_PRIVILEGE	INSUFFICIENT_PRIVILEGE
CONNECT_PROBLEM	SYS_ERROR
INIT_PROBLEM	SYS_ERROR
DBS_NOT_CONNECTED	SYS_ERROR
DBS_STOPPED	SYS_ERROR
DISCONNECT_PROBLEM	SYS_ERROR
SESSION_NOT_OPEN	SYS_ERROR
INACCESSIBLE_FILES	SYS_ERROR
FILE_SYSTEM_FULL	SYS_ERROR
POOL_SATURATED	SYS_ERROR
TABLE_SPACE_FULL	SYS_ERROR
UNKNOWN_PRINTER	SYS_ERROR
FILE_NOT_PRINTABLE	SYS_ERROR
SPOOLER_DISACTIVATED	SYS_ERROR
PRINTER_NOT_DEFINED	SYS_ERROR
NO_PRINTER_AVAILABLE	SYS_ERROR
UNKNOWN_ERROR	SYS_ERROR
NOT_ENOUGH_SPACE	SYS_ERROR
NOT_CONNECTED_TO_EVAL	SYS_ERROR
FA_INTERNAL_PROBLEM	SYS_ERROR
FA_UNIX_PROBLEM	SYS_ERROR
FA_MMI_TIME_OUT	SYS_ERROR
FA_LIST_NOT_FOUND	SYS_ERROR
LIST_OVERFLOW	SYS_ERROR
COMMS_TIME_OUT	SYS_ERROR
UNKNOWN_PRIVILEGE	SYS_ERROR

DBS_ERROR	TEV_ERROR
DBS_ORACLE_PROBLEM	SYS_ERROR
UNIX_PROBLEM	SYS_ERROR
DBS_INTERNAL_PROBLEM	SYS_ERROR
DBS_COMMUNICATION_PROBLEM	SYS_ERROR
BAD_SELECTION_CRITERIA	SYS_ERROR
EXISTING_FA_JOB	SYS_ERROR
FA_DRIVE_NOT_AVAILABLE	SYS_ERROR

D-5.1.2 TEV Footer messages

As described earlier in section D-5.1 the footer message in combination with the status may report an error. The footer message denotes the action to be performed, the status message says wether the action was successful or not and specifies the nature of the error. Corrective actions depend on the displayed combination of message and status.

The following messages may be reported in TEV footers :

Start TEV

Quit TEV

Start to initialise session

Start Events Logging Tool

Start Raw Data Dump Tool

Start Data Set Generation Tool

Start Statistics Generation Tool

Start Data Listing Generation Tool

Start Graph Tool

Loading a Definition

Saving the Current Definition

Updating the Definition

Printing the Current Definition

Resetting the Current Definition

Displaying the Current Definition

Executing the Current Definition

Printing a Definition

Getting the Definition List

Deleting a Definition

Clearing Definition

Overwriting a Definition

Loading the Current Result

Saving the Current Result

Printing the Current Result

Displaying the Current Result

Printing a Result

Getting the Result List

Storing a Result

Renaming a Result

Copying a Result
Overwriting a Result
Deleting a Result
Getting the Parameter List
Selecting Parameters
Checking the Parameter Set
Inserting a Parameter
Formatting a Packet
Setting the output parameters
Fetching the Current Packet
Jumping to a Packet
Saving the Current Packet
Overwriting a Packet
Printing the Current Packet
Displaying the Current Packet
Selecting an Evaluation Session
Creating an Evaluation Session
Deleting an Evaluation Session
Confirm deletion of an Evaluation Session
Getting Evaluation Session List
Getting Evaluation Session Info
Selecting an Execution Session
Creating an Execution Session
Deleting an Execution Session
Initialising Execution Sessions
Confirm deletion of an Execution Session
Getting Execution Session List
Getting Execution Session Info
Getting Data Set Info
Merging Data Set
Start Merge
Getting User Event List
Selecting User Events
List CCUS From MDB
Getting the online data

Getting archived sessions

Getting on line sessions

Archiving a session

Retrieving a session

Deleting on line data

Getting session data location

Deleting session data

Updating final archive window

Converting a Result

Importing a session

Exporting a session

Updating Import/Export window

Getting job list

Removing job

Selecting default printer

Displaying job queue

Displaying SAS tool

Selecting a SAS

SAS Selected

Starting a SAS

D-5.1.3 TEV Status Messages

As described earlier in section D-5.1 the footer message in combination with the status may report an error. The footer message denotes the action to be performed, the status message says wether the action was successful or not and specifies the nature of the error. Corrective actions depend on the displayed combination of message and status.

The following TEV status are reported with the message described.

TEV_STATUS	MESSAGE
OK	”Ok.”
RUNNING	”...”
TEV_INTERNAL_ERROR	”TEV Internal error.”
DBS_ERROR	”DBS error.”
MDB_ERROR	”MDB error.”
SYS_ERROR	”System error”
PRINT_ERROR	”Print error.”
ACCESS_VIOLATION	”Access violation.”
TOOLS_OVERFLOW	”Tools overflow.”
TOOLS_RUNNING	”Tools are running.”
LIST_OVERFLOW	”List Overflow.”
EXISTING_USER	”Existing User.”
TRUNCATED_LIST	”Truncated List.”
FILE_NOT_EXISTING	”File not existing.”
FILE_IS_EMPTY	”File is empty.”
FILE_NOT_SELECTED	”No file selected.”
NO_DIRECTORY_SELECTED	”No directory selected.”
DIRECTORY_NOT_EXISTING	”Directory not existing.”
DIRECTORY_ALREADY_EXISTING	”Directory already existing.”
NO_TARGET_FILE_SELECTED	”Target file not selected.”
NO_SOURCE_FILE_SELECTED	”Source file not selected.”
SOURCE_FILE_UNDEFINED	”Undefined source file.”
DESTINATION_UNDEFINED	”Undefined destination.”
FILE_ALREADY_EXISTING	”File already existing.”
TOO_LONG_FILENAME	”Filename is too long.”
TOO_LONG_DIRNAME	”Directory name is too long.”
FILE_NOT_PRINTABLE	”File not printable.”
FILE_SYSTEM_FULL	”File system full.”

FILE_ACCESS_ERROR	”File access error.”
NOT_AN_ARCHIVE_FILE	”Invalid archive file.”
INVALID_SOURCE	”Invalid source.”
TOO_MANY_SEL_EXECUTION_SESSIONS	”Too many selected execution sessions.”
INVALID_SESSION_NAME	”Invalid session name.”
NO_EXECUTION_SESSION_SELECTED	”No execution session selected.”
NO_EVALUATION_SESSION_SELECTED	”No evaluation session selected.”
SESSION_NOT_FOUND	”Session not found.”
SESSION_IS_OPEN	”Session is open”
SESSION_IS_ALLOCATED	”Session is allocated”
SESSION_IS_USED	”Session is used”

A SESSION_IS_USED error will be reported if a user tries to delete, archive or export a session which is not exclusively allocated to him.

SESSION_IS_EMPTY	”Session is empty”
SESSION_IS_ARCHIVED	”Session is archived”
SESSION_NOT_ARCHIVED	”Session is not archived”
SESSION_IS_ONLINE	”Session is online”
SESSION_ALREADY_EXISTING	”Session already existing.”
JOB_CANCELLED	”Job cancelled.”
DATA_ALREADY_ONLINE	”Data already online.”
DATA_PARTIALLY_ONLINE	”Data partially online.”
DATA_NOT_ONLINE	”Data not online.”
PRINTING_REROUTED	”Printing re–routed.”
CANCEL_NOT_AUTHORISED	”Cancel not authorised.”
INSUFFICIENT_PRIVILEGE	”Insufficient privilege.”
FA_ERROR	”Final archive error.”
EMPTY_LIST	”List is empty.”
DATA_NOT_FOUND	”Data not found.”
NULL_SESSION_NAME	”Session name is null.”
WRONG_SELECTION	”Wrong selection.”
INVALID_CCU	”Invalid CCU.”
NO_INITIAL_TIME_FRAME	”Initial time frame not defined.”
NO_GLOBAL_TIME_FRAME	”Global time frame not defined.”
NULL_TIME_FRAME	”Time frame is null.”
TIME_FRAME_ERROR	”Error in time frame.”

TIME_FRAME_NOT_IN_INITIAL	”Time frame not included in initial time frame.”
TIME_FRAME_NOT_IN_OVERALL	”Time frame not included in overall time frame.”
INVALID_DATE_TIME	”Invalid date and time.”
LEXICAL_ERROR_IN_TIME_FRAME	”Lexical error in time frame.”
PATHNAME_ERROR	”Pathname error.”
INVALID_PARAMETER	”Parameter is invalid.”
DUPLICATE_PARAMETER	”Parameter is duplicated.”
INVALID_SOFTWARE_PARAMETER	”Software parameter is invalid.”
NO_PARAMETER_SELECTED	”No parameter selected.”
EMPTY_PARAMETER_SET_REMOVED	”An empty parameter set has been removed.”
NO_SOURCE_SELECTED	”Source is not selected.”
DATA_SET_GENERATION_ERROR	”Error during generation of data set.”
CALIBRATION_ERROR	”Calibration error.”
ARCHIVE_FILE_ERROR	”Archive file error.”
SAMPLING_ERROR	”Sampling error.”
SID_ERROR	”SID error.”
TOO_MANY_PARAMETERS	”Too many parameters.”
NO_MORE_PACKET	”No more packet.”
ADU_NOT_EXISTING	”ADU not found.”
GDU_NOT_EXISTING	”GDU not found.”
DATA_ERROR	”Data Error.”
NO_CURRENT_ADU	”No current ADU.”
NO_CURRENT_GDU	”No current GDU.”
INCOMPLETE_PACKET_TYPE	”Incomplete packet type.”
INVALID_OUTPUT_FORMAT	”Invalid output format.”
OUT_OF_RANGE_PACKET_SIZE	”Packet size is out of range.”
WRONG_BYTES_PER_LINE	”Wrong number of bytes per line.”
EXISTING_DEFINITION	”Existing definition.”
NO_DEFINITION_EXISTING	”No existing definition.”
NO_DEFINITION_SELECTED	”No definition selected.”
NOT_EXISTING_DEFINITION	”Not existing definition.”
NO_CHANGE	”No change.”
NULL_DEFINITION	”Null definition.”
NULL_DEFINITION_NAME	”Null definition name.”
ERROR_ACCESSING_DEFINITION	”Error accessing definition.”

EXISTING_RESULT	”Existing result.”
RESULT_GENERATION_ERROR	”Result generation error.”
RESULT_TYPE_UNDEFINED	”Result type undefined.”
NO_RESULT_EXISTING	”No result existing.”
NO_RESULT_SELECTED	”No result selected.”
NOT_COMPLETE_RESULT	”Not complete Result.”
NOT_EXISTING_RESULT	”Not existing result.”
NULL_RESULT	”Null result.”
NULL_RESULT_NAME	”Null result name.”
NOT_COMPLIANT_PARAMETER_LIST	”Not compliant parameter list.”
TRUNCATED_PARAMETER_LIST	”Truncated parameter list.”
MISMATCHED_PARAMETER_TYPES	”Mismatched parameter types.”
NOT_EXISTING_DATA_SET	”Not existing data set.”
NO_DATA_SET_SELECTED	”Not data set selected”
INSUFFICIENT_PARAMETERS_DEFINED	”Insufficient parameters have been defined”
LEXICAL_ERROR_IN_SCALING_STR	”Lexical error in graph scaling string”
INVALID_SCALING	”Error in graph scaling”
INVALID_NAME	”Error in graph name”
UNABLE_TO_CONNECT_TO_NEWS	”Unable to connect to NeWS server”
TOO_MANY_SESSIONS	”Merged data set would contain too many sessions”
SOURCE_MISMATCH	”Sets to merge contain parameter from different sources”
ENGINEERING_UNIT_MISMATCH	”Sets to merge contain parameter with different units”
VALUE_MISMATCH	”Sets to merge have value discrepancies”
INVALID_DATA_SET	”Invalid data set.”
INSUFFICIENT_USER_EVENTS_SELECTED	”Insufficient user events selected.”
OVERLAPPING_TIME_FRAME	”Overlapping time frame.”
DIFFERENT_CCUS	”Selected sessions have different CCUs.”
MERGE_SELECT_ERROR	”Data sets to merge must be selected on same time type”
MERGE_ORDER_ERROR	”Data sets to merge must be ordered by same time type”
NO_JOB_SELECTED	”Job not selected”
SAS_NOT_EXISTING	”SAS not existing”
SAS_NOT_SELECTED	”SAS not selected”
SAS_NOT_EXECUTABLE	”SAS not executable”

D-5.1.4 TEV Error Messages

TEV error messages may come from different sources. In the following sections TEV error messages are listed with the corresponding error text of the source which reported the error.

**** MAIN PROGRAM ABANDONED — EXCEPTION ”TEV_PROCESS_LOCKED” RAISED**
—> Verify no other TEV is running on this node. Then remove lock file: rm /tmp/TEV.lock

ERROR (TDCS_RPI) INIT_TDCS_RPI: Exception raised
FATAL_ERROR (TDCS_RPI) Can't initialize TDCS RPI
—> There is a problem with the TDCS (SVF specific) software
—> can be ignored on environments where SVF is not installed

ERROR : TEV Internal error.Current exception: X_ERROR_ACCESSING_FILE
Consequence : Current operation fails with unexpected error.
Recovery action : No specific recovery. See system administrator.
Debug information : EXECUTE_CURRENT_DEFINITION”);

ERROR : System error.
Consequence : Current operation fails.
Recovery action : Modify for specific problem reported in cause.
Debug information : TEV_MAIN_MENU:TASK_TO_QUIT_TEV_CB”);

GENERATE_DATA_SET > TEV Internal error. > Exception MDB_ERROR raised.
—> There is a problem in the MDB. Verify messages on command line in window where
—> TEV (resp. Task Selector) was started: Are there any further messages explaining
—> the reason on the command line ?

PRINT_RESULT > System error. > UNKNOWN_PRINTER
—> The printer indicated in the environment variables VICOS_PRINTER_n was not
—> found on VICOS_PRINT_SERVER
—> Are the names set in $DBS_HOME/user_env/dbs_cshrc correct ?
—> Is a printer with this name installed and known to the node where TEV is running ?

1. QUIT_DBS_ACCESS > System error. > TDCS Error Status: TDCS_NOT_CONNECTED
2. TEV_MAIN_MENU:TASK_TO_QUIT_TEV_CB > System error. >
—> Is TDCS running ?
—> can be ignored on environments where SVF is not installed

Unix error : EACCES in LOCAL_PRINT_SERVICES SEND_FOR_PRINTING.
Permission denied PRINT_RESULT > System error. > DBS_UNIX_PROBLEM
—> can probably be ignored
—> file is put into print queue
—> DBS seems to have a problem when checking the status

TEV_MAIN_MENU:TASK_TO_QUIT_TEV_CB > TEV Internal error. >

QUIT_MDB_ACCESS > TEV Internal error. > Current Exception: USE_ERROR

E NOTATIONAL CONVENTIONS

<table>
<tr><th>Notation</th><th>Description</th></tr>
<tr><td>left button
middle button
right button</td><td>The Buttons are the three buttons on the mouse. The names (Left, Middle, Right) refer to their position relative to the others. The names appear in bold style.</td></tr>
<tr><td>click / hold /
double click
< left | middle | right ></td><td>These indicate the action to carried out using the left, middle or right mouse buttons.
Click means a short touching of the indicated mouse button.
The double click means a fast double touching of the mouse button.
Hold means pressing of the mouse button and releasing it after selection.</td></tr>
<tr><td>–></td><td>A menu path is indicated by arrow (e.g. File–>Open). To execute a command in a menu or (series of) sub-menu hold down the indicated mouse button and drag the mouse to the right.</td></tr>
<tr><td>< keys | commands ></td><td>The names of keys and commands are shown in bold style</td></tr>
<tr><td>menu | box names</td><td>Names of menus and boxes are identified in bold and italic style.</td></tr>
<tr><td><variable></td><td>Variables will appear in italic style enclosed in ’< >’.</td></tr>
<tr><td><code>text</code></td><td>Text in courier font identifies text which <u>must</u> be entered exactly as shown</td></tr>
<tr><td></td><td>The hand is used when something important must be noted.
The information is written in italic.</td></tr>
<tr><td>▶</td><td>The triangle is used to identify a procedure with one ore more procedural steps.</td></tr>
<tr><td>■</td><td>The box is used to identify procedural steps. The steps <u>must</u> be executed by the user.</td></tr>
<tr><td></td><td>The shadowed box is used to identify optional procedural steps, which can be executed by the user.</td></tr>
<tr><td>•</td><td>Bullets list provided information, but <u>not</u> procedural steps.</td></tr>
<tr><td>[doc. number]</td><td>References to documents are enclosed in square brackets .</td></tr>
</table>

F DESCRIPTION OF THE CGS TES API INTERFACE

The CGS Test Execution Software (TES) Application Programming Interface (API) comprises a set of functions as interface to SAS. Those functions are described in detail in the following sub–sections.

Additional functions can be found in the package ADT_TES_TO_SAS_COMMAND not described in this appendix. The package contains helpful procedures which work on a lower, more detailed level.
The procedures provide functions e.g. to put parameter or message strings into a command, to set an ADU description in a command, or – vice versa – to extract parameters from a command.

The TES API comprises the following main functions ;

– Setting the polling rate (obsolete for CGS 4.2.0)
– Connecting to and disconnecting from CGS
– Reading and handling commands
– Handling GDUs
– Handling ADUs
– Reading enditem data from CGS
– Providing enditem data to CGS
– Exchanging messages with APs
– Downloading software
– Reading the CGS time
– Reporting errors to CGS

F-1 Setting the polling rate (obsolete)

Note: This paragrah is only valid for CGS Version up to CGS 4.1.1.x. In CGS 4.2.0, the polling has been replaced by signal handling to increase performance

Before a SAS connects to CGS, the polling rate can be configured. This must not be done explicitly, but then a default polling rate applies.

Configuring the polling rate has the advantage that the CPU consumption of a SAS and the UNIX system load of the host computer running it is significantly reduced. Normally, a SAS will check for incoming commands every 10 milliseconds. This is necessary to ensure fast reaction times. This in turn results in a high UNIX process scheduler load because the SAS has to be scheduled for execution every 10 msec. However, not in all cases looking for commands that frequent is useful.

If a SAS produces only ADUs for example, then a lower polling frequency of 200 msec is more adequate. A reaction within 200 msec in response to an ADU request is by far fast enough, normally.

If a SAS processes GDUs, a high polling rate is normally desirable, because ”immediate” reactions are required as a response to the user’s request for sending a telecommand or stimulus.
To define the polling interval in a SAS, the operation

SET_POLLING_INTERVAL

must be called. This operation has one parameter, being a positive integer number which determines the polling interval in terms of milliseconds. E.g. a call to TES_API.SET_POLLING_INTERVAL(300) will tell the TES_API to look for incoming commands every 300 milliseconds.

If the parameter is not supplied or if the procedure SET_POLLING_INTERVAL is not called before CONNECT, a default polling interval of 10 milliseconds applies.

F-2 Connecting to and disconnecting from CGS

Before a SAS can start communication with CGS it has to connect to CGS first. For SAS running on an HP computer, the CONNECT operation will connect the SAS to the local instance of TES on the same HP the SAS is running on. For SAS running on a SUN, the host name of the parent TES has to be provided as a parameter to the connect operation.

The mechanism to connect to CGS is the operation

CONNECT

from the TES_API package.

CONNECT has to be called fast after the SAS is started since TES has a built–in time–out mechanism which will reject SAS CONNECT requests after the elapsed time. The time interval between calling the UCL system library call to start the SAS and the SAS then calling CONNECT via the TES API is configurable through the TES config file.

If the polling rate for incoming commands shall be adjusted, then the operation SET_POLLING_INTERVAL has to be called before the CONNECT operation (see above).

Inside the SAS, the CONNECT operation can be called at any place before and/or after any internal initialisation or processing which has to be performed.

The syntax of the procedure to connect to CGS is

```
procedure CONNECT (
        SAS_NAME          : in     VICOS_DEFINITIONS.T_APPLICATION_NAME;
        CGS_PARENT        : in     VICOS_DEFINITIONS.T_APPLICATION_NAME :=
                                   VICOS_DEFINITIONS.EMPTY_NAME_STRING
        Application_ID    : out    TES_DEFINITIONS.T_APPLICATION_ID;
        STATUS            : out    TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The SAS_NAME is the name of the executable image of the SAS. This is required since TES needs to correlate the name of an application started through a UCL library command with the ones trying to connect. It is of type T_APPLICATION_NAME from the package VICOS_DEFINITIONS. This parameter has to be assigned carefully (and correctly !) since otherwise the API will return with error code 99 (OTHER_ERROR). A possible assignment looks as follows:
  ```
  SAS_NAME: constant VICOS_DEFINITIONS.T_APPLICATION_NAME :=(”SAS_TMTC            ”,8);
  ```
  The T_APPLICATION_NAME is a record with two components, one being the name of the SAS (a twenty character string, filled with blanks, the SAS name being filled in left justified) and a length field(giving the number of valid characters for the SAS name, excluding the blanks!).

– The CGS_PARENT is the application name of the parent TES, e.g. ”TES_01”, as defined in the System Topology Table. The CGS_PARENT denotes the process logical name where the SAS shall connect to. This parameter must only be supplied in case the SAS runs on a node different from the local node. i.e. only, if the TES to connect to runs on a different node than the SAS itself.
A common mechanism to transfer the CGS_PARENT to the SAS is via the start parameter the user can give in the LOAD_APPLICATION command. The SAS can read these parameter as command line parameter and use the value given for CGS_PARENT in the CONNECT procedure.

– The Application_ID is returned by TES. It is the unique handle to be used in all future calls to the TES–API for this specific instance of TES.

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k.
CURRENT_MODE_WRONG	TES is in idle or error mode or it is currently not running
COMMUNICATION_ERROR	some communication problem exists, check system topology table
SAS_UNKNOWN	The SAS_NAME is wrong, this SAS has not been started by the CGS_PARENT
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

After all processing within the SAS has finished, the SAS has to disconnect from CGS. This is done using the procedure

DISCONNECT

from the TES_API package. After this procedure has been called, no further communication is possible with CGS and the SAS should terminate itself. Usually, this is done in V2 by calling the UNIX _exit system service as the last executable statement. (Because not all tasks inside the TES_API can be terminated, this construct has to be used, otherwise the process could not stop due to the fact that the Ada run–time system would assume that some tasks are still active and would not allow termination). In V3 this will be changed TBD.

The syntax of the procedure to disconnect from CGS is

```
procedure DISCONNECT (Application_ID : in      TES_DEFINITIONS.T_APPLICATION_ID;
                      STATUS         : out     TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
NOT_CONNECTED	the SAS is not yet connected
COMMUNICATION_ERROR	some communication problem exists
INVALID_APPLICATION_ID :	The identifier is wrong, perhaps an old one is used?
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

F-3 Reading and handling commands

In this section the general mechanisms of handling commands from CGS will be explained. More information can be found in the subsequent sections where the processing of certain data is described in more detail.

A SAS has to react according to commands received from TES, normally. The mechanism to accept such commands is the procedure

READ_COMMAND

from the TES_API package. This procedure delivers exactly one command to the SAS in a blocking or non–blocking mode.

The syntax of the procedure to read commands from CGS is

```
procedure READ_COMMAND
        (APPLICATION_ID   : in      TES_DEFINITIONS.T_APPLICATION_ID;
        COMMAND   : in out ADT_TES_TO_SAS_COMMAND.T_COMMAND;
        COMMAND_ID        : out TES_DEFINITIONS.T_IDENTIFIER;
        STATUS            : out     TES_DEFINITIONS.T_RETURN_STATUS;
        BLOCK             : in      BOOLEAN := FALSE);
```

The parameters of this operation have the following meaning:

– APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– COMMAND is the command read (see below for an explanation of how to deal with this command.

– COMMAND_ID is a unique identifier for this command to be used in all subsequent replies to CGS with respect to this command, e.g. an acknowledge.

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is not in NORMAL mode. Check test node status display
COMMUNICATION_ERROR	some communication error occurred. Check system topology table.
INVALID_APPLICATION_ID	The identifier is wrong, perhaps an old one is used?
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

– BLOCK is indicating whether the procedure shall block until a command is sent from CGS or not. In case the parameter is set to false, the procedure will not block and will always return, either with a command or with no command. In case it is set to true, then the procedure will only return after a command has been sent to the SAS.

Types of Commands

The following types of commands can be sent from CGS to a SAS:

– UNDEFINED
This type of command indicates some error in the communication link to CGS and this command should normally not appear. In this case the SAS should send an error message to CGS.

– INIT_APPLICATION
This command is sent whenever the CGS user wants to initialise a SAS through a call to the corresponding UCL system library procedure (INIT_APPLICATION). Together with the command, an initialisation command (string) can be passed to the SAS and it is up to the SAS to perform the proper initialisation then. CGS does not check that a SAS has been initialised nor does it require it. Consequently, it is up to the user or the SAS programmer to define whether a SAS needs explicit initialisation or not.

– START_APPLICATION
This command is sent whenever the CGS user wants to start the SAS processing internally through a call to the corresponding UCL system library procedure (START_APPLICATION). This does not mean starting a SAS as an operating system process but it means internal starting of the SAS after it has been stopped, e.g. by a RESET command.. CGS does not check that a SAS has been started nor does it require it. Consequently, it is up to the user or the SAS programmer to define whether a SAS needs explicit starting or not.

– RESET_APPLICATION
This command is sent whenever the CGS user wants to reset the SAS processing internally through a call to the corresponding UCL system library procedure (RESET_APPLICATION). This does not mean resetting a SAS as an operating system process but it means internal resetting of the SAS, e.g. after internal errors have been detected. CGS does not check that a SAS has been reset nor does it require it. Consequently, it is up to the user or the SAS programmer to define whether a SAS needs explicit resetting or not.

– STATUS_REQUEST
This command is sent whenever the CGS user wants to read the SAS status through a call to the corresponding UCL system library procedure (GET_APPLICATION_STATUS). The SAS has to respond to this command by supplying the current status (see below)

– AP_MESSAGE
This command indicates a message has been sent from an automated procedure to the SAS through the UCL system library procedure SEND_MESSAGE_TO_APPLICATION. The SAS can extract the message from the command and process it.

– ADU_REQUEST
This command indicates to the SAS that CGS wants a certain ADU to be dispatched from now on or that a previous request for dispatching an ADU shall now be cancelled. The corresponding ADU description as stored in the mission database is passed to the SAS together with this command.

– SUPPLY_ADU
This command contains the ADU supplied by CGS, as previously requested by this SAS. ADU re–routing to SASes will be implemented in CGS V3 only.

– GDU_REQUEST
This command indicates to the SAS that a certain telecommand or stimulus shall be generated to the unit under test (or the test system itself). The GDU supplied as part of the command contains all necessary information in order for the SAS to generate the telecommand/stimulus to the corresponding front end hardware.

– SW_DOWNLOAD
This command indicates to the SAS that some Software replaceable unit from the mission database has to be downloaded into the unit under test or some front end equipment.

– UNLOAD_APPLICATION
  This command is sent to the SAS as a result of the UCL system library procedure UNLOAD_APPLICATION. The SAS shall then disconnect from TES, stop its internal processing and terminate as an operating system process.

– NO_COMMAND
  This ”command” is returned by the procedure READ_COMMAND whenever there is no command to read!

Command Acknowledgement

Commands have to be acknowledged by the SAS in a special way. For this purpose, the procedure

ACKNOWLEDGE_COMMAND

from the TES_API package has to be called.

The syntax of the procedure to acknowledge a command from CGS is

```
1. procedure ACKNOWLEDGE_COMMAND_WITH_STATUS (
         APPLICATION_ID     : in      TES_DEFINITIONS.T_APPLICATION_ID;
         COMMAND_ID         : in      TES_DEFINITIONS.T_IDENTIFIER;
         SAS_STATUS         : in      TES_DEFINITIONS.T_LINK_STATUS;
         SAS_ERRORS         : in      INTEGER;
         STATUS             : out     TES_DEFINITIONS.T_RETURN_STATUS;
         SAS_LAST_ERROR     : in      STRING := ””);

2. procedure ACKNOWLEDGE_COMMAND
         (APPLICATION_ID    : in      TES_DEFINITIONS.T_APPLICATION_ID;
         COMMAND_ID         : in      TES_DEFINITIONS.T_IDENTIFIER;
         ACKNOWLEDGED       : in      BOOLEAN;
         STATUS             : out     TES_DEFINITIONS.T_RETURN_STATUS);

3. procedure ACKNOWLEDGE_COMMAND
         (APPLICATION_ID    : in      TES_DEFINITIONS.T_APPLICATION_ID;
         COMMAND_ID         : in      TES_DEFINITIONS.T_IDENTIFIER;
         ACKNOWLEDGED       : in      BOOLEAN;
         ACK_CODE           : in      INTEGER;
         ACK_LOT            : in      CGS_CALENDAR.CGS_DATE_AND_TIME
                            := CGS_CALENDAR.NULL_DATE_AND_TIME;
         ACK_LOT_NANOSEC    : in      INTEGER := 0;
         ACK_SMT            : in      CGS_CALENDAR.CGS_DATE_AND_TIME
                                      := CGS_CALENDAR.NULL_DATE_AND_TIME;
         ACK_SMT_NANOSEC    :  in     INTEGER := 0;
         STATUS             : out     TES_DEFINITIONS.T_RETURN_STATUS);
```

As can be seen, three versions of this procedure exist for the following purposes:
– reply to the STATUS_REQUEST command
– simple acknowledge or advanced acknowledge of all other commands

The parameters of this operation have the following meaning:

– The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– COMMAND_ID is the identifier for this command returned by procedure READ_COMMAND.

- ACKNOWLEDGED is the boolean indicating to CGS whether the SAS acknowledges the command (TRUE) or not (FALSE).
- ACK_CODE identify more precisely the success or type of error occurred when executing the command. The valid range is 1000 .. 5000. **Be careful**: This code is routed to UCL/HLCL level instead of Ground_Library.SUCCESS.
- ACK_LOT identify precisely when the command has been executed (based on LOT).
- ACK_LOT_NANOSEC define additional time information.
- ACK_SMT identify precisely when the command has been executed (based on SMT).
- ACK_SMT_NANOSEC define additional time information.
- SAS_STATUS is the current status of the SAS. It is and Ada enumeration type with the following alternatives:

RESET	a RESET_APPLICATION command has reset the SAS
INITIALISED	an INIT_APPLICATION command has initialised the SAS
RUNNING	a START_APPLICATION command has internally started processing, the SAS is running normally
ABORTED	the SAS has internally detected an error and has stopped processing, reset. re–initialisation or re–starting is needed possibly

- SAS_ERRORS is a simple counter value in which the SAS can return the cumulative number of errors internally detected. This value is not checked by CGS.
- SAS_LAST_ERROR is the text of the last error message or a descriptive text of the last error detected by the SAS.
- STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is not in NORMAL mode. Check test node status display
COMMUNICATION_ERROR	some communication error occurred. Check system topology table.
INVALID_APPLICATION_ID	The identifier is wrong, perhaps an old one is used?
INVALID_COMMAND_ID	The identifier is wrong, perhaps an old one is used?
COMMAND_TIMEOUT	The acknowledge comes too late, sorry...
COMMAND_NEEDS_NO_ACK	: The command needs no acknowledge
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

How a specific type of command has to be acknowledged can be seen from the following list:

- UNDEFINED
  This type of command needs no acknowledgement at all.

- INIT_APPLICATION
  This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.

- START_APPLICATION
  This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.

- RESET_APPLICATION
  This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.

- STATUS_REQUEST
  This command has to be acknowledges by calling alternative 1 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details. All relevant SAS status data have to be provided to CGS.

- AP_MESSAGE
  This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.

- ADU_REQUEST
  This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.

- SUPPLY_ADU
  This type of command needs no acknowledgement at all.

– GDU_REQUEST
  This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.
– SW_DOWNLOAD
  This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.
– UNLOAD_APPLICATION
  This type of command needs no acknowledgement at all.
– NO_COMMAND
  This type of command needs no acknowledgement at all.

Command parsing

The commands returned from CGS by procedure READ_COMMAND are private types in the Ada implementation sense. As such, the programmer can only parse them using the package ADT_TES_TO_SAS_COMMAND which provides the corresponding type definition in order to create Ada variables of type T_COMMAND plus all procedures to extract the relevant information from the command.

The following information is provided in a command from CGS to SAS:

– COMMAND_ALTERNATIVE
  this is an enumeration type with the following possible values:
  UNDEFINED
  INIT_APPLICATION
  START_APPLICATION
  RESET_APPLICATION
  STATUS_REQUEST
  AP_MESSAGE
  ADU_REQUEST
  SUPPLY_ADU
  GDU_REQUEST
  SW_DOWNLOAD
  UNLOAD_APPLICATION
  NO_COMMAND
  The list of possible values exactly corresponds to the set of commands which can be sent from CGS to a SAS.
– COMMAND_TIMEOUT
  A positive number indicating in milliseconds the allowed time interval between the reception of the command and the acknowledge to CGS.

– STRING_LENGTH
  In case an INIT_APPLICATION or AP_MESSAGE command is received this figure indicates the number of characters associated with the command as the message string.

– INIT_PARAMS
  In case of an INIT_APPLICATION command, the initialisation string provided to the SAS through the UCL system library procedure. The length of this string can be obtained separately

– MESSAGE
  In case of an AP_MESSAGE command, the message text sent from the UCL system library procedure to the SAS. The length of this string can be obtained separately

– ADU_DESCRIPTION
  In case of an ADU_REQUEST command, the ADU description of the ADU to be supplied later by this SAS.

– ADU
  In case of a SUPPLY_ADU command, the ADU routed by CGS.

– GDU
  In case of a GDU_REQUEST command, the GDU to be processed by this SAS, i.e. forwarded to the unit under test in the form of a telecommand or stimulus

– DATABASE_SW_REPLACEABLE_UNIT
  In case of a SW_DOWNLOAD command, the SW to be downloaded into the unit under test or front end equipment.

Ada Command Handling Example

In the following, an example piece of Ada code is given which shows the general logic to be applied when parsing and handling commands received from CGS:

```
with TES_DEFINITIONS;                  -- This package provides some common types
with TES_API;                          -- this package provides the TES_API operations !
with ADT_TES_TO_SAS_COMMAND;           -- this package provides a high level interface for
...                                    --  handling the commands sent from TES to SAS
procedure MAIN is                      -- here the main program starts
CMD: ADT_TES_TO_SAS_COMMAND.T_COMMAND;-- the variable for the command
READ_STATUS: TES_DEFINITIONS.T_RETURN_STATUS; -- the return status of API operations
procedure SEND_ACK_TO_CGS(...) is      -- a general procedure to acknowledge commands
...
end SEND_ACK_TO_CGS;
...
begin
...                                    -- do some initialisations
loop                                   -- the main program loop starts here
...                                    -- do some more initialisations
TES_API.READ_COMMAND(...,              -- wait for a command and block
                     COMMAND => CMD,
                     ...,
                     STATUS => READ_STATUS,
                     BLOCK => TRUE);
```

```
if READ_STATUS /= TES_DEFINITIONS.SUCCESS then -- check if READ_COMMAND was successful
   TEXT_IO.PUT_LINE ( ”Error during reading command from TES”);
else
                                       -- now parse the command using Ada CASE
case ADT_TES_TO_SAS_COMMAND.COMMAND_ALTERNATIVE(COMMAND_FROM_TES) is
when ADT_TES_TO_SAS_COMMAND.INIT_APPLICATION =>
   -- the INIT_APPLICATION_COMMAND
   TEXT_IO.PUT_LINE(”INIT_APPLICATION Command”); -- process the INIT_APPLICATION command
   SEND_ACK_TO_CGS(COMMAND_ID);        -- acknowledge the command
 when ADT_TES_TO_SAS_COMMAND.START_APPLICATION=>
   -- the START_APPLICATION_COMMAND
        TEXT_IO.PUT_LINE(”START_APPLICATION command”);
        SEND_ACK_TO_CGS(COMMAND_ID);
...                                    -- more types of commands
end if                                 -- the check for a valid READ_COMMAND
end loop                               -- the main program loop ends here
end MAIN;                              -- here the main program ends
```

F-4 Handling GDUs

GDUs are sent from CGS to the SAS using the GDU_REQUEST command. Before that happens, the SAS has to indicate to CGS though that it is able to handle GDUs. For this purpose, the procedure

ANNOUNCE_GDU_SERVICE

from the TES_API has to be called.
The syntax of this operation is as follows:

```
procedure ANNOUNCE_GDU_SERVICE(
         APPLICATION_ID    : in      TES_DEFINITIONS.T_APPLICATION_ID;
         STATUS            : out     TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is idle or error mode or it is current not running
COMMUNICATION_ERROR	some communication error occurred. Check system topology table.
INVALID_APPLICATION_ID	The identifier is wrong, perhaps an old one is used?
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

After this operation has been called, all GDUs meant for this SAS will be sent to it by CGS whenever necessary, i.e. when a UCL system library command ISSUE is encountered or when a monitoring exception occurs.

As soon as the SAS is no longer in a position to process GDUs, it can call the operation

WITHDRAW_GDU_SERVICE

from the TES_API.
The syntax of this operation is as follows:

```
procedure WITHDRAW_GDU_SERVICE(
         APPLICATION_ID    : in      TES_DEFINITIONS.T_APPLICATION_ID;
         STATUS            : out     TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is idle or error mode or it is current not running
COMMUNICATION_ERROR	some communication error occurred. Check system topology table.
INVALID_APPLICATION_ID	The identifier is wrong, perhaps an old one is used?
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

After this operation has been called, CGS will refuse to send those GDUs meant for this SAS and will inform the user accordingly through appropriate error messages.

A SAS may temporarily change over its local status (ANNOUNCE_GDU_SERVICE, WITHDRAW_GDU_SERVICE, ...) at any time and more than once during a 'session' with CGS.

When a GDU_REQUEST command is then read from CGS, the GDU is obtained from that command using the corresponding function in the ADT_TES_TO_SAS_COMMAND. This GDU can be processed internally in the SAS, sent to the corresponding front end equipment and from there to the unit under test. At any point in time the SAS is free to send the acknowledge for this command back to CGS.

It is important to note here that it is the responsibility of the SAS programmer/designer to decide, at which point in time the acknowledge for a GDU will be given to CGS. The two extreme cases are:

– The acknowledge is given immediately after the command has been read and it has been decided that reading the command was successful, or

– The command is read, the stimulus is sent to the unit under test and only after an expected reaction has been seen from the unit under test or the front end equipment then the acknowledge is given to CGS

Whatever the SAS designer/programmer decides he/she must be aware of the fact that CGS checks for the predefined timeout interval when sending out a GDU to a SAS, i.e. before that interval expires the ACK/NACK must have been received by CGS.

<u>GDU Processing</u>

Similar to the commands sent from CGS to SAS, the GDUs are also private types, encapsulated in an Ada package. This package provides the type definition for the T_GDU plus all operations to read the information needed.

The following information is part of the GDU and available to the SAS through proper use of the ADT_GDU:

– ID
The database short identifier (SID) of this GDU_(DESCRIPTION). The GDU_DESCRIPTION is the database contents describing the GDU received from CGS.

– ALTERNATIVE
  There are four GDU alternatives defined currently in CGS:

ANALOG_STIMULUS	a stimulus with an analogue parameter, e.g. SET_TEMPERATURE(VALUE) where SET_TEMPERATURE denotes the stimulus and VALUE is the actual value.
DISCRETE_STIMULUS	a fully defined stimulus, e.g. POWER_ON
PREDEFINED_TC	a CCSDS telecommand
UNDEFINED	an undefined type (not used)

– VALUE_OF
  In case of an ANALOG_STIMULUS, the actual value of the parameter. In case of a DISCRETE_STIMULUS the value is an integer derived from the STATECODE value supplied as a parameter on UCL level

– PRIORITY
  This indicates the priority with which the GDU has been handled inside CGS. It should be good practice to handle GDUs with similar priority inside the SAS also.
  CGS uses three levels of priority:
  NORMAL: the lowest priority
  HIGH: all GDUs with high priority interrupt the sending of NORMAL priority GDUs
  EMERGENCY: these GDUs are generated as a result of monitoring exceptions and have highest priority.

– TIME_TAG
  The point in time when this GDU shall really be sent to the unit under test.
  IMPORTANT : If the GDU contains a CCSDS packet, the time tag field of the GDU can always be ignored by the SAS since the time tag actually supplied when sending the GDU/CCSDS packet is put into the secondary header of the CCSDS packet.

– SEQUENCE_COUNT
  In case of CCSDS predefined TCs, the sequence count of the CCSDS packet being contained in this GDU

– RETRIES
  The number of retries defined for this GDU in the database

– PHYSICAL_ADDRESS

The low level device specific information needed by the SAS to really issue the GDU. This information is stored in the configuration database and passed to the SAS by CGS. CGS does not interpret it. It is up to the SAS to decide, how the information in the physical address field should be used. It is, however, good practice to re–use as much from the information contained herein, in order to ensure consistency enforced by the use of the configuration database also in the SAS area and in order to simplify reconfiguration of front end equipments by simple MDB modifications without having to re–code the SAS.

The information contained in the physical address field of the GDU is the following:

NODE_ADDRESS	this field contains a string identifying the name of the physical device which is responsible for handling this command
DEVICE_TYPE	this field contains a string identifying the type of device to be used for generating this kind of output to the unit under test, can be used for consistency checking
DEVICE_ADDRESS	this field contains a string identifying the device specific address to be used to ’speak’ to the respective device via a dedicated bus, e.g. the device address on a given IEEE488 bus or the RT address on a given MIL–bus
DEVICE_SUBADDRESS	this field contains a string identifying the device specific subaddress to be used to ’speak’ to the respective device via a dedicated bus, e.g. the device subaddress on a given IEEE488 bus or the RT subaddress on a given MIL–bus
CHANNEL	this field contains an integer number identifying the channel to which the respective command shall be issued, i.e. typically a channel of an IEEE488 device or an output channel on a dedicated MIL–bus RT.
FUNCTION_CODE	this field contains an integer number identifying the function code to be sent to the specific channel in order to generate the desired output. The function code can be mapped internally to function commands in the SAS, if desired.
PROTOCOL_ID	this field contains an integer number identifying the kind of protocol to be handled between the SAS and the front end device, e.g. ”IEEE488 serial poll” or ”MIL–Bus mode command with data”
CMD1	this field contains a string identifying the command to be sent to the front end device in order to generate the desired output.
CMD2	this field contains a string identifying another command to be sent to the front end device in order to generate the desired output.

– PREDEFINED_TC

For the GDUs of type PREDEFINED_TC, the CCSDS packet to be sent to the unit under test.

Using the procedures and functions of the ADT_GDU properly, the stimuli to the unit under test or the front end devices can be generated.

Time–tagged GDUs

CGS provides the possibility to time–tag GDU, i.e. they shall be sent at a specific point in time. This feature must, however, be implemented inside the SAS sending the GDU last but not least. As such, part of the GDU sent as a command from CGS to SAS is the time tag specifying at which date and time it has to be sent.

IMPORTANT : see above comments concerning the time tag field in case of CCSDS packets!

It is the responsibility of the SAS to manage GDU queues to ensure that GDUs are sent out in the correct order and at the correct point in time.

Retries of GDUs

Together with the GDU sent out as a command from CGS to the SAS, a re–try parameter is passed to the SAS, indicating how often the SAS shall re–try to send the GDU in case of transmission errors.

It is the responsibility of the SAS to manage GDU re–try queues to ensure that GDUs are sent out in the correct order and with the correct number of re–tries

GDU handling example

In the following an example for handling a CCSDS type GDU is given. This example extracts the CCSDS packet from the GDU and sends it to a front end device.

```
with TES_DEFINITIONS;                 –– the list of withed packages
with VICOS_DEFINITIONS;
with ADT_GDU;
WITH ADT_GDU_DESCRIPTION;
with ADT_CCSDS_PACKET;
with NUMERIC_TYPES;
with UNCHECKED_CONVERSION;
with TEXT_IO;
...
package GDU_MANAGER is                –– specify a package for handling GDUs

task type T_GDU_MAIN is               –– use a task to handle GDUs
entry init(AP_ID: in TES_DEFINITIONS.T_APPLICATION_ID);
entry send_GDU(GDU : in ADT_GDU.T_GDU);
entry fini;
end T_GDU_MAIN;

for T_GDU_MAIN'storage_size use 400_000;       –– set the stack size, may be necessary for
                                                   big GDUs, 400_000 is just a figure

HANDLER : T_GDU_MAIN;                          –– declare a TASK HANDLER

end GDU_MANAGER;                               –– end of the package specification
```

```
package body GDU_MANAGER is                     -- the body of the package

global_ap_id : TES_DEFINITIONS.T_APPLICATION_ID;-- some global variables
global_gdu : ADT_GDU.T_GDU;
global_gdu_alternative : ADT_GDU_DESCRIPTION.T_GDU_ALTERNATIVES;
global_ccsds : ADT_CCSDS_PACKET.T_CCSDS_PACKET;
global_send_it : boolean := false;
global_out_line : TEXT_IO.file_type;
global_out_filename : string(1..11) := "/dev/tty0p6";

task body T_GDU_MAIN is
begin
accept init(AP_ID: in TES_DEFINITIONS.T_APPLICATION_ID) do
         global_ap_id := AP_ID;
         text_io.open(global_out_line,text_io.out_file,global_out_filename);
         ...                -- do some more initialisations
end init;


loop
select
accept   send_gdu(GDU : in ADT_GDU.T_GDU) do
         global_gdu_alternative := adt_gdu.alternative(GDU);-- which kind of GDU
         if (global_gdu_alternative = CCSDS_PACKET) then
         global_ccsds := ADT_GDU.predefined_tc(GDU);   -- extract the CCSDS packet
         len := ADT_CCSDS_PACKET.user_data_length(global_ccsds);-- the length
         send_CCSDS_PACKET(global_ccsds,len,out_file);
         ...                                           -- do error checks etc.
         end if;                     -- the CCSDS packet
end send_gdu;
or
accept fini do
         text_io.close(global_out_line);
end fini;
end select;
end loop;
end T_GDU_MAIN;
end GDU_MANAGER;
```

F-5 Handling ADUs

ADUs are sent from SAS to CGS. Before that happens, the SAS has to indicate to CGS though that it is able to handle ADUs. For this purpose, the procedure

ANNOUNCE_ADU_SERVICE

from the TES_API has to be called.
The syntax of this operation is as follows:

```
procedure ANNOUNCE_ADU_SERVICE(
         APPLICATION_ID     : in     TES_DEFINITIONS.T_APPLICATION_ID;
         STATUS             : out    TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is idle or error mode or it is current not running
COMMUNICATION_ERROR	some communication error occurred. Check system topology table.
INVALID_APPLICATION_ID	The identifier is wrong, perhaps an old one is used?
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

After this operation has been called, CGS will eventually send requests for certain ADUs to the SAS, typically as a consequence of the UCL system library command START_ACQUISITION.

As soon as the SAS is no longer in a position to process ADUs, it can call the operation

WITHDRAW_ADU_SERVICE

from the TES_API.
The syntax of this operation is as follows:

```
procedure WITHDRAW_ADU_SERVICE(
         APPLICATION_ID     : in     TES_DEFINITIONS.T_APPLICATION_ID;
         STATUS             : out    TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is idle or error mode or it is current not running
COMMUNICATION_ERROR	some communication error occurred. Check system topology table.
INVALID_APPLICATION_ID	The identifier is wrong, perhaps an old one is used?
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

After this operation has been called, CGS will no longer send new requests for ADUs to the SAS and will inform the user accordingly through appropriate error messages in case certain ADUs would be needed.

A SAS may temporarily change over its local status (ANNOUNCE_ADU_SERVICE, WITHDRAW_ADU_SERVICE, ...) at any time and more than once during a 'session' with CGS.

When a ADU_REQUEST command is then read from CGS, the ADU_DESCRIPTION is obtained from that command using the corresponding function in the ADT_TES_TO_SAS_COMMAND. This ADU_DESCRIPTION can be processed internally in the SAS. At any point in time the SAS is free to send the acknowledge for this command back to CGS.

It is important to note here that it is the responsibility of the SAS programmer/designer to decide, at which point in time the acknowledge for an ADU_REQUEST will be given to CGS. The two extreme cases are:

– The acknowledge is given immediately after the command has been read and it has been decided that reading the command was successful, or

– The command is read, the acquisition of all data related to the ADU_REQUEST can be done because it has been checked that all front end devices are o.k.

Whatever the SAS designer/programmer decides he/she must be aware of the fact that CGS checks for the predefined timeout interval when sending out an ADU_REQUEST to a SAS, i.e. before that interval expires the ACK/NACK must have been received by CGS.

ADU_REQUEST Processing

Similar to the commands sent from CGS to SAS, the ADU_DESCRIPTION being part of the ADU_REQUEST is also private type, encapsulated in an Ada package (The ADU_DESCRIPTION can be extracted from the command using an operation of the ADT_TES_TO_SAS_CMD). This package provides the type definition for the T_ADU_DESCRIPTION plus all operations to read the information needed.

The following information is part of the ADU_DESCRIPTION and available to the SAS through proper use of the ADT_ADU_DESCRIPTION:

– DB_SID
  The database short identifier (SID) of this ADU_DESCRIPTION. The ADU_DESCRIPTION is the database contents describing the ADU to be sent to CGS.

– PRIVATE_ID
This is an ASCII string defined in the database by the user. Using the private ID, the chicken–egg problem of what has to be defined first, the SAS or the database, can be avoided. The SAS programmer can write the SAS using the private ID to identify which kind of ADU to generate in response to this ADU_REQUEST. Also, the database can be filled with ADU_DESCRIPTION at any time using a predefined set of private IDs. At no point in time must an SAS programmer know the SID of a dedicated ADU_DESCRIPTION. This is extremely important, if the SAS programmer decided not to interpret the ADU_REQUEST with respect to the individual measurements contained in it but instead the SAS generates a predefined datapacket.

– ALTERNATIVE
There are four ADU alternatives defined currently in CGS:

STRUCTURED	a structured ADU already contains a list of CGS compatible raw values in the CGS raw value data format. This kind of ADU is the preferred one if the SAS reads individual values of individual enditems from individual front ends.
UNSTRUCTURED	in this case the SAS reads a binary data buffer from a front end and simply puts the whole buffer into an ADU. Subsequently, VICOS will unpack the data from the binary buffer, do the raw value extraction and convert the bits and pieces into VICOS compatible raw values for further processing.
CCSDS_PACKET	a CCSDS telemetry packet available as a binary data packet in the SAS. Subsequently, VICOS will unpack the data from the CCSDS packet, do the raw value extraction and convert the bits and pieces into VICOS compatible raw values for further processing.
UNDEFINED	an undefined type (not used)

– ACQUISITION_RATE
The acquisition rate in terms of milliseconds for the ADU to be provided to CGS. 1000 indicates the ADU shall be sent every second. A value of 0 indicates that the ADU shall be sent whenever possible, i.e. typically this is the value for a CCSDS packet since it is unknown when a specific packet will be received on ground. CGS itself does not check the acquisition frequency of an SAS.

– GLOBAL_ADDRESS
The low level device specific information needed by the SAS to really acquire the data for the corresponding ADU from the font end devices. This information is stored in the configuration database and passed to the SAS by CGS. CGS does not interpret it. It is up to the SAS to decide, how the information in the physical address field should be used. It is, however, good practice to re–use as much from the information contained herein, in order to ensure consistency enforced by the use of the configuration database also in the SAS area and in order to simplify reconfiguration of front end equipments by simple MDB modifications without having to re–code the SAS.
The information contained in the physical address field of the ADU_DESCRIPTION is the following:

NODE_ADDRESS	this field contains a string identifying the name of the physical device which is responsible for handling this command
DEVICE_TYPE	this field contains a string identifying the type of device to be used for

	generating this kind of input from the unit under test, can be used for consistency checking
DEVICE_ADDRESS	this field contains a string identifying the device specific address to be used to ’speak’ to the respective device via a dedicated bus, e.g. the device address on a given IEEE488 bus or the RT address on a given MIL–bus
DEVICE_SUBADDRESS	this field contains a string identifying the device specific subaddress to be used to ’speak’ to the respective device via a dedicated bus, e.g. the device subaddress on a given IEEE488 bus or the RT subaddress on a given MIL–bus
CHANNEL	this field contains an integer number identifying the channel from which the respective data shall be read, i.e. typically a channel of an IEEE488 device or an input channel on a dedicated MIL–bus RT.
FUNCTION_CODE	this field contains an integer number identifying the function code to be sent to the specific channel in order to generate the desired input. The function code can be mapped internally to function commands in the SAS, if desired.
PROTOCOL_ID	this field contains an integer number identifying the kind of protocol to be handled between the SAS and the front end device, e.g. ”IEEE488 serial poll” or ”MIL–Bus mode command with data”
CMD1	this field contains a string identifying the command to be sent to the front end device in order to generate the desired input.
CMD2	this field contains a string identifying another command to be sent to
the	
	front end device in order to generate the desired input.

The GLOBAL_ADDRESS information must not be present in all ADU_DESCRIPTIONs. It only makes sense, if all data belonging to the related ADU can be acquired from one device, one subaddress and one channel. This is the case for the unstructured ADUs and the CCSDS–packet ADUs since the data buffer (the CCSDS packet) typically comes from exactly one location. In the other case, each individual enditem has its private physical address information (see below)

– ENDITEM_DESCRIPTION
This information identifies which measurement enditems defined in the mission database have to be put into this ADU. The following type of information is comprised in the ADU_DESCRIPTION:

SID	the short identifier of the respective measurement
RAW_VALUE_TYPE	the identification of the VICOS provided raw value type, which in fact is an enumeration type with the following alternatives: UNDEFINED SINGLE_BIT GROUP_OF_BITS INTEGER FLOAT BYTE_STREAM This information is only available for structured ADUs since only then the SAS has to put in raw values
INDEX	The index position of this raw value in the list of raw values of a structured ADU, e.g. 10 indicates this raw value is the tenth in the list.
PARAMETER_ADDRESS	the physical address information specifying how to obtain the data. The contents of this field is the same as for the GLOBAL_ADDRESS field of the ADU_DESCRIPTION.

– CCSDS_ID
This field indicates the CCSDS application ID of the CCSDS packet to be encapsulated in this ADU.

Providing ADUs to CGS

The processing of an ADU includes the typical processing steps, after having read the ADU_REQUEST command and extracted the ADU_DESCRIPTION from it:

– determine the type of ADU to be provided by checking the PRIVATE_ID of the ADU_DESCRIPTION

– establish an Ada variable holding the ADU by using the CONSTRUCT operation from ADT_ADU with the ADU_DESCRIPTION as an input

– start the acquisition of data from the front end equipment(s) for those data to be put into the ADU

– fill the data into the ADU by using the appropriate operation from ADT_ADU:

SET_RAW_VALUE:	adds an individual raw value to the ADU. Individual raw values can be added until the list of required values has been completed
SET_BINARY_BUFFER:	puts the binary buffer into the ADU
SET_CCSDS_PACKET:	puts the CCSDS packet into the ADU

– set the time tag of the ADU using the operation SET_TIME_TAG of the ADT_ADU. The time tag should carry the actual time (local + SMT) when the raw data have been acquired. For CCSDS packets the time tag from the packet should be extracted.

– set the ADU sequence counter using the operation SET_SEQUENCE_COUNT from ADT_ADU. All ADUs have a sequence counter which has to be set and maintained by the SAS.

– set the data interruption flag of the ADU using the operation SET_DATA_INTERRUPTION from ADT_ADU. All ADUs have a data interruption indication which is normally set to false, i.e. no data has been lost between two subsequent ADUs. Should the SAS detect, however, that between two subsequent ADUs there has been a data interruption (for whatever reason, e.g. a CCSDS packet has been lost, a device has not responded in time, some raw values were bad,...) the first ADU after the data interruption should be flagged with DATA_INTERRUPTION set to TRUE.

– send the ADU to CGS at the predefined time interval using the procedure
SUPPLY_ADU
from the TES_API package.
The syntax of this operation is as follows:

```
procedure SUPPLY_ADU(
        APPLICATION_ID    : in      TES_DEFINITIONS.T_APPLICATION_ID;
        ADU               : in out  ADT_ADU.T_ADU;
        STATUS            : out     TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– ADU is the ADU to be supplied

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is idle or error mode or it is current not running
COMMUNICATION_ERROR	some communication error occurred. Check system topology table.
INVALID_APPLICATION_ID	The identifier is wrong, perhaps an old one is used?
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

Suspension of ADU delivery

In the case where the application is no longer able to provide a given ADU to CGS, it can indicate that the ADU delivery is suspended. This will lead to have all mesaurements from the ADU to become the status STATIC, until the delivery resumes. A specific operation allows to indicate the suspension of the delivery, while it is resumed simply by calling the next time SUPPLY_ADU for that ADU. The syntax of that operation is as follows:

```
procedure SUSPEND_ADU (
        APPLICATION_ID    : in      TES_DEFINITIONS.T_APPLICATION_ID;
        ADU_SID           : in      MPS_DEFINITIONS.SID;
        STATUS            : out     TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– ADU is the suspended ADU

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is idle or error mode or it is current not running
COMMUNICATION_ERROR	some communication error occurred. Check system topology table.
INVALID_APPLICATION_ID	The identifier is wrong, perhaps an old one is used?
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

ADU handling example

In the following example, an ADU handling example is shown. The SAS waits for an ADU request and then sends a predefined ADU of type CCSDS packet without parsing the ADU description contained in the ADU request. The input data from the front end is assumed to be a binary block.

```
with TES_DEFINITIONS;                -- the list of withed packages
with VICOS_DEFINITIONS;
with CGS_CALENDAR;
with ADT_GDU;
with ADT_ADU;
with ADT_ADU_DESCRIPTION;
with ADT_CCSDS_PACKET;
with CALENDAR;
with NUMERIC_TYPES;
with TES_API;
with UNCHECKED_CONVERSION;
with TEXT_IO;


package ADU_MANAGER is               -- define a package for ADU handling

task type T_ADU_MAIN is              -- define a task type for ADU handling
entry init(AP_ID: in TES_DEFINITIONS.T_APPLICATION_ID);
entry insert_adu(ADU : in ADT_ADU_DESCRIPTION.T_ADU_DESCRIPTION);
entry delete_adu(ADU : in ADT_ADU_DESCRIPTION.T_ADU_DESCRIPTION);
entry fini;
end T_ADU_MAIN;

for T_ADU_MAIN'storage_size use 400_000;-- set the stack size, 400_000 is just an example

HANDLER : T_ADU_MAIN;                -- define a task HANDLER

end ADU_MANAGER;                     -- end of the package spec.
```

```
package body ADU_MANAGER is

   max_dde_i_length : integer := 255;
   packet_size : integer := 1024;
  subtype T_UDATA_BUFFER is NUMERIC_TYPES.byte_array(1..packet_size);
  subtype T_DDE_INPUT_BUFFER is string(1..max_dde_i_length);
  subtype T_BA4 is NUMERIC_TYPES.BYTE_ARRAY(1..4);
  timetag : CGS_CALENDAR.T_DATE_AND_TIME;
  global_ap_id : TES_DEFINITIONS.T_APPLICATION_ID;
  global_adu_description : ADT_ADU_DESCRIPTION.T_ADU_DESCRIPTION;
  global_send_it : boolean := false;
  global_ccsds : ADT_CCSDS_PACKET.T_CCSDS_PACKET;
  udata : T_UDATA_BUFFER := (others => 0);
  global_adu : ADT_ADU.T_ADU;
  adu_seq_nb : integer := 1;
  rtn : TES_DEFINITIONS.T_RETURN_STATUS;
  global_delay : duration;
  dde_data : T_DDE_INPUT_BUFFER;
  act_dde_len, acq_rate_in_msec : natural;


  task body T_ADU_MAIN is
   begin
    accept init(AP_ID: in TES_DEFINITIONS.T_APPLICATION_ID) do
            global_ap_id := AP_ID;
                  -- set up the CCSDS packet here
            ADT_CCSDS_PACKET.set_packet_type(global_ccsds,ADT_CCSDS_PACKET.tm);
           ADT_CCSDS_PACKET.set_application_id(global_ccsds,41);
           ADT_CCSDS_PACKET.set_sequence_count(global_ccsds,1);
           ADT_CCSDS_PACKET.set_length(global_ccsds,packet_size);
          . . .
           ADT_CCSDS_PACKET.set_packet_checksum_type(global_ccsds,
                                                  ADT_CCSDS_PACKET.no_checksum);
           timetag := CGS_CALENDAR.get_date_and_time(CALENDAR.CLOCK);
           ADT_CCSDS_PACKET.set_time_tag(global_ccsds,timetag);
           ADT_CCSDS_PACKET.set_user_data(global_ccsds,udata);
    end init;
   loop
   select
   accept  insert_adu(ADU : in ADT_ADU_DESCRIPTION.T_ADU_DESCRIPTION) do
                  acq_rate_in_msec := ADT_ADU_DESCRIPTION.ACQUISITION_RATE(ADU);
                  global_send_it := true;
                  ADT_ADU.construct(global_adu,ADU); -- create the ADU
          end insert_adu;
   or
     accept   delete_adu(ADU : in ADT_ADU_DESCRIPTION.T_ADU_DESCRIPTION) do
                  TEXT_IO.PUT_LINE("Removing an ADU");
                  global_send_it := false;
          end delete_adu;
    else
          delay(global_delay);
```

```
                -- read data from the front end device here
         if global_send_it then
              interprete_input_data(udata); -- udata block returned
             ADT_CCSDS_PACKET.set_user_data(global_ccsds,udata); -- put block into  CCSDS
             timetag := CGS_CALENDAR.get_date_and_time(CALENDAR.CLOCK);
             ADT_CCSDS_PACKET.set_time_tag(global_ccsds,timetag);
             ADT_ADU.set_time_tag(global_adu,timetag);
             ADT_ADU.set_sequence_number(global_adu,adu_seq_nb);
             adu_seq_nb := adu_seq_nb + 1;
             ADT_ADU.set_ccsds_packet(global_adu,global_ccsds);
             TES_API.supply_adu(global_ap_id,global_adu,rtn);
             if integer(rtn) /= integer(TES_DEFINITIONS.SUCCESS) then
                  TEXT_IO.PUT_LINE("After sending ADU, RTN="&
                           TES_DEFINITIONS.T_RETURN_STATUS'image(rtn));
            end if;
         end if;
end select;

end loop;

end T_ADU_MAIN;

end ADU_MANAGER;
```

F-6 Reading enditem data from CGS

SASes have the possibility of reading individual enditem data from CGS. Raw data as well as engineering data can be provided. This mechanisms simplifies the access to individual enditem data in case the SAS only seldom needs rare enditem data.

CGS delivers raw and engineering data to SASes out of its internal data pool, i.e. the most recent value is delivered. However, the delivery of individual raw data enditem values to SAS is not synchronised with the delivery of ADUs by SAS. A raw value is always delivered immediately out of the data pool to the SAS if it is currently available, that is it has been acquired before and the acquisition has not been stopped in the mean time. The similar principle applies also for delivering engineering data

The procedure to read a raw data value from CGS is the

READ_RAW_VALUE

operation from the TES_API.
The syntax of this operation is as follows:

```
procedure READ_RAW_VALUE(
         APPLICATION_ID    :in       TES_DEFINITIONS.T_APPLICATION_ID;
         ENDITEM_PATHNAME  :in       string;
         VALUE_ALTERNATIVE : out     VICOS_DEFINITIONS.T_RAW_VALUE_ALTERNATIVES;
         INT_VALUE         : out     NUMERIC_TYPES.INTEGER32;
         UINT_VALUE        : out     NUMERIC_TYPES.UNSIGNED_INTEGER32;
         FLOAT_VALUE       : out     NUMERIC_TYPES.SINGLE_FLOAT;
         BYTESTREAM_VALUE  : in  out MPS_DEFINITIONS.DYNAMIC_STRING;
         STATUS            : out     TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– ENDITEM_PATHNAME is the enditem pathname of the raw value. The enditem has to refer to an EGSE_xxx_MEASUREMENT in the database since only the measurements have raw values.

– VALUE_ALTERNATIVE indicates the type of the enditem value to be read, e.g. whether it is an INTEGER or a FLOAT.

– INT_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is integer, in other cases this value is undefined;

– UINT_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is unsigned integer, in other cases this value is undefined;

– FLOAT_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is float, in other cases this value is undefined;

– BYTESTREAM_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is byte stream, in other cases this value is undefined;

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is in idle or error mode or it is currently not running
COMMUNICATION_ERROR	some communication problem exists, carefully look into your system topology table, maybe TES died in the meantime
INVALID_APPLICATION_ID	The identifier is wrong, perhaps an old one is used?
ENDITEM_NOT_ACQUIRED	the enditem is currently not acquired, so there is no raw value
ENDITEM_UNKNOWN	the enditem is unknown, so the pathname is wrong or the item is maintained on another test node.
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

The procedure to read an engineering value from CGS is the

READ_ENGINEERING_VALUE

operation from the TES_API.
The syntax of this operation is as follows:

```
procedure READ_ENGINEERING_VALUE(
         APPLICATION_ID     : in      TES_DEFINITIONS.T_APPLICATION_ID;
         ENDITEM_PATHNAME   : in      string;
         VALUE_ALTERNATIVE  : out     VICOS_DEFINITIONS.T_RAW_VALUE_ALTERNATIVES;
         INT_VALUE          : out     NUMERIC_TYPES.INTEGER32;
         UINT_VALUE         : out     NUMERIC_TYPES.UNSIGNED_INTEGER32;
         FLOAT_VALUE        : out     NUMERIC_TYPES.SINGLE_FLOAT;
         STATECODE_VALUE    : out     MPS_DEFINITIONS.STATE_CODE;
         BYTESTREAM_VALUE   : in out  MPS_DEFINITIONS.DYNAMIC_STRING;
         STATUS             : out     TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– ENDITEM_PATHNAME is the enditem pathname of the value. A reference to an enditem of type EGSE_xxx_MEASUREMENT or EGSE_xxx_SW_VARIABLE is allowed. The enditem is identified through its database pathname. For simplicity reasons this is a string

– VALUE_ALTERNATIVE indicates the type of the enditem value to be read, e.g. whether it is an INTEGER or a FLOAT.

– INT_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is integer, in other cases this value is undefined;

– STATECODE_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is STATE_CODE, in other cases this value is undefined;

– FLOAT_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is float, in other cases this value is undefined;

– BYTESTREAM_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is byte stream, in other cases this value is undefined;

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is in idle or error mode or it is currently not running
COMMUNICATION_ERROR	some communication problem exists, carefully look into your system topology table, maybe TES died in the meantime
INVALID_APPLICATION_ID	The identifier is wrong, perhaps an old one is used?
ENDITEM_NOT_ACQUIRED	the enditem is currently not acquired, so there is no raw value
ENDITEM_UNKNOWN	the enditem is unknown, so the pathname is wrong or the item is maintained on another test node.
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

A SAS may read any raw or engineering value available in a distributed CGS test system configuration. CGS will internally manage to obtain the value from the place where it is maintained.

F-7 Providing enditem data to CGS

SASes can write into SW variables maintained by CGS. SW variable have to be defined in the configuration database. For this purpose, the operation

WRITE_ENGINEERING_VALUE

from the TES_API has to be used.
The syntax of this operation is as follows:

```
procedure WRITE_ENGINEERING_VALUE (
         APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
         ENDITEM_Pathname  : in     STRING;
         VALUE             : in     NUMERIC_TYPES.INTEGER32;
         STATUS            : out    TES_DEFINITIONS.T_RETURN_STATUS);
procedure WRITE_ENGINEERING_VALUE (
         APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
         ENDITEM_Pathname  : in     STRING;
         VALUE             : in     NUMERIC_TYPES.UNSIGNED_INTEGER32;
         STATUS            : out    TES_DEFINITIONS.T_RETURN_STATUS);
procedure WRITE_ENGINEERING_VALUE (
         APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
         ENDITEM_Pathname  : in     STRING;
         VALUE             : in     NUMERIC_TYPES.SINGLE_FLOAT;
         STATUS            : out    TES_DEFINITIONS.T_RETURN_STATUS);
procedure WRITE_ENGINEERING_VALUE (
         APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
         ENDITEM_Pathname  : in     STRING;
         VALUE             : in     MPS_DEFINITIONS.STATE_CODE;
         STATUS            : out    TES_DEFINITIONS.T_RETURN_STATUS);
```

```
procedure WRITE_ENGINEERING_VALUE (
        APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
        ENDITEM_Pathname  : in     STRING;
        VALUE             : in     MPS_DEFINITIONS.DYNAMIC_STRING;
        STATUS            : out    TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of these operations have the following meaning:

- The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.
- ENDITEM_PATHNAME is the enditem pathname of the value. A reference to an enditem of type EGSE_xxx_SW_VARIABLE is allowed.
- VALUE is the engineering value to be written CGS. The overloaded versions of the procedure WRITE_ENDITEM_VALUE have different types for the formal parameter VALUE depending on the type of value to be written into the enditem. Care has to be taken that the correct type of enditem data is written into an enditem, otherwise exceptions will occur
- STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is in idle or error mode or it is currently not running
COMMUNICATION_ERROR	some communication problem exists, carefully look into your system topology table, maybe TES died in the meantime
INVALID_APPLICATION_ID	The identifier is wrong, perhaps an old one is used?
ENDITEM_NOT_ACQUIRED	the enditem is currently not acquired, so there is no raw value
ENDITEM_UNKNOWN	the enditem is unknown, so the pathname is wrong or the item is maintained on another test node.
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)
TYPE_MISMATCH	the enditem is not of type SW variable.
TYPE_MISMATCH	the enditem's type defined in the database and the data supplied do not match.
NOT_A_SW_VALUE	the enditem exists but is not a SW variable

When a SAS writes to a SW variable, the monitoring of that variable will be done by CGS afterwards (if defined so in the configuration database).

F-8 Exchanging messages with APs

SAS can read messages from APs and can also send messages to them. Messages are simple ASCII strings.

When an AP wants to send a message to a SAS it uses a UCL system library procedure. The result of calling this procedure is that a command is sent to the SAS, containing the message as a string. Of course, the SAS must be running at the point in time the message is sent to it since they are not buffered.

At any point in time the user can also call the corresponding UCL system library procedure from the HLCL command window and thus send a message to a SAS.

The procedure for sending a message from a SAS to an automated procedure is

SEND_MESSAGE_TO_AP

from the TES_API.
The syntax of this operation is as follows:

```
procedure SEND_MESSAGE_TO_AP(
         APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
         AP_NAME           : in     string;
         MESSAGE           : in     TES_DEFINITIONS.T_AP_MESSAGE;
         MESSAGE_LENGTH    : in     positive;
         STATUS            : out    TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.
– AP_NAME is the pathname of the automated procedure to which the message shall be sent. This AP must be running on the local test node which this SAS logically ”belongs” to. If more than one instance of the AP is running, the message will be passed to all APs.
– MESSAGE is the message to be written to the AP. The type definition in TES_DEFINITIONS enforces a length of 80 characters for the message (left justified, filled with blanks if message is shorter).
– MESSAGE_LENGTH is the actual length of the message.
– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is in idle or error mode or it is currentl not running
COMMUNICATION_ERROR	some communication problem exists, carefully look into your system topology table, maybe TES died in the meantime
INVALID_APPLICATION_ID :	The identifier is wrong, perhaps an old one is used?
AP_NOT_FOUND	The AP does not exist on the testnode
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

It is good programming practice that an AP should pass its name to the SAS it wants to exchange messages with and then the SAS should only use those AP names previously registered.

F-9 Downloading software

In case a SAS shall download Software, a special command will be sent to it containing the software or a reference to it.

After it has been determined that a command received by a SAS is a DOWNLOAD command (using function COMMAND_ALTERNATIVE of ADT_TES_TO_SAS_COMMAND), using procedure

```
 procedure GET_DOWNLOAD_PARAMS (
         OF_COMMAND        : in out  T_COMMAND;
```

```
        FILE_TYPE         : out    T_DOWNLOAD_SW_TYPE;
        FILE_ID           : out    VICOS_DEFINITIONS.T_NAME_STRING;
        FILE_NAME         : out    VICOS_DEFINITIONS.T_FILE_NAME;
        DESTINATION_TYPE  : out    T_DESTINATION;
        DESTINATION_ID    : out    VICOS_DEFINITIONS.T_NAME_STRING);
```

of ADT_TES_TO_SAS_COMMAND the attributes of the item to be downloaded can be determined. The fields have the following meaning:

– OF_COMMAND
  This is the reference of the command received, usually a variable holding the command

– FILE_TYPE
  This is a variable which holds a value determined by an enumeration type definition. Its purpose is to differentiate between the type of information to be downloaded. The possible values can be:
  DATA_FILE
  EXECUTABLE
  according to what has been defined in MDB for this download item (must be an enditem in MDB of type EGSE_SOFTWARE). It is up to the application to determine whether or not the value of this field has a meaning for the SAS.

– FILE_ID
  This field is a private identifier for the item to be downloaded. It is stored in MDB together with the absolute UNIX pathname (or SWEU reference). Thus a logical name for concrete items can be defined and absolute UNIX pathnames needn't be hard–coded in SAS. This is useful if a SAS needs to perform special actions in depending on the item to be downloaded. Then using the logical name, the SAS can determine how to handle the item.

– FILE_NAME
  The absolute UNIX filename of the item to be downloaded.

– DESTINATION_TYPE
  This is a variable which holds a value determined by an enumeration type definition. Its purpose is to differentiate between the destinations for downloading. The possible values can be:
  FRONT_END_EQUIPMENT
  UNIT_UNDER_TEST
  according to what has been defined in MDB for this download item (must be an enditem in MDB of type EGSE_SOFTWARE). It is up to the application to determine whether or not the value of this field has a meaning for the SAS.

– DESTINATION_ID
  Is a private identifier for the destination for downloading. The SAS may use a simple table look up operation to determine what is the concrete target from the local name provided here.

F-10 Reading the CGS time

The CGS concept foresees that all computer clocks of all machines involved in a test are synchronised by the Time Synchronisation Software (TSS) to an accuracy of 1..2 msec. The computer clocks run the local time (wall clock). A SAS could therefore use the standard Ada mechanisms to obtain the local time. Also, delays based on local time can be achieved in this way. In parallel to the local time, CGS maintains a simulated mission time (SMT) which is under user control. At any point in time the SMT can be stopped, started or reset. Once it is running, it runs with a constant offset in parallel to the local time.

TES_API provides one operation to read the local time plus the simulated mission time SMT. This operation is called

READ_TIME

in the TES_API.
The syntax of this operation is as follows:

```
procedure READ_TIME(
         APPLICATION_ID     : in      TES_DEFINITIONS.T_APPLICATION_ID;
         LOCAL_TIME         : out     CGS_CALENDAR.T_DATE_AND_TIME;
         SMT                : out     CGS_CALENDAR.T_DATE_AND_TIME;
         SMT_RUNNING        : out      BOOLEAN;
         STATUS             : out     TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– LOCAL_TIME is the local time in the CGS_CALENDAR layout, i.e. three integer numbers specifying the year, month and day plus four integer numbers indicating the hours, minutes, seconds and milliseconds.

– SMT is the simulated mission time in the CGS_CALENDAR layout, i.e. three integer numbers specifying the year, month and day plus four integer numbers indicating the hours, minutes, seconds and milliseconds.The SMT may return a special value (CGS_CALENDAR.NULL_DATE_AND_TIME) in case it is not initialised.

– SMT_RUNNING indicates, if the SMT is counting or stopped.

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
TSS_ERROR	the time synchronisation sw could not provide timing information for unknown reasons

TES_API provides another operation to read the details of the SMT. This operation is called

READ_SMT_DETAILS

in the TES_API.
The syntax of this operation is as follows:

```
procedure READ_SMT_DETAILS (
        APPLICATION_ID   : in      TES_DEFINITIONS.T_APPLICATION_ID;
        SMT              :    out CGS_CALENDAR.CGS_DATE_AND_TIME;
        SYNC_SW_STATUS   :    out BOOLEAN;
         SYNC_SW_LOCKED   :    out BOOLEAN;
         SYNC_SW_IN_LIMIT :    out BOOLEAN;
        SMT_SERVER       :    out BOOLEAN;
         MTU_PRESENT      :    out BOOLEAN;
         SMT_STATUS       :    out T_SMT_STATUS;
        SMT_OFFSET       :    out INTEGER;
         MTU_STATUS       :    out INTEGER;
         STATUS           :    out TES_DEFINITIONS.T_RETURN_STATUS);
```

This procedure returns the detailed status of the SMT.

– The application id is the one returned by TES during the connect operation.

– SYNC_SW_STATUS returns, whether the NTP Status file, which is updated every, minute by a crontab job if NTP is running, was read and has the correct format

– SYNC_SW_LOCKED returns, whether synchronisation source for the local NTP process is considered as reliable.

– SYNC_SW_IN_LIMIT returns, whether the time deviation between the NTP client and its synchronisation source is in the allowed limit

– SMT_SERVER returns, whether the local machine is SMT server (TRUE) or not

– MTU_PRESENT returns, whether a MTU is present (TRUE) or not

– SMT_STATUS returns the actual SMT status

– SMT_OFFSET returns the actual SMT offset

– MTU_STATUS returns the actual MTU status

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	the call was successfull
TSS_ERROR	the time synchronisation sw could not provide timing information for unknown reasons

Should the SAS want to delay until a given local time or SMT, the operation

WAIT_UNTIL

from the TES_API has to be used.
The syntax of this operation is as follows:

```
procedure WAIT_UNTIL(
         APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
         WAKE_UP_TIME      : in     CGS_CALENDAR.T_DATE_AND_TIME;
         TIME_TYPE         : in     TES_DEFINITIONS.TIME_TYPE
         STATUS            : out    TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– WAKE_UP_TIME is the time in the CGS_CALENDAR layout, i.e. three integer numbers specifying the year, month and day plus four integer numbers indicating the hours, minutes, seconds and milliseconds at which the SAS shall be activated again.

– TIME_TYPE is an enumeration type allowing the two alternatives LOCAL_TIME and SMT only.

– STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
SMT_RESET	the SMT has been reset so the wake up time may not be correct.
SS_ERROR	the time synchronisation sw could not provide timing information for unknown reasons

This procedure will block until the desired wake–up time has expired and then return successfully (normally). Should an SMT wait–until be desired and should the SMT be reset in the meantime, the procedure will return immediately after the reset of the SMT.

F-11 Reporting errors and passing messages to CGS

At any point in time the SAS might detect errors and / or might want to pass a message to CGS. For this purpose, the procedure

SEND_ERROR_MESSAGE

in the TES_API has to be used.
Messages sent from a SAS to CGS will be displayed in the CGS error window and they will be logged in the test result database.
NOTE: For historic reasons the procedure is called SEND_ERROR_MESSAGE although a SAS can also send normal information messages..
The syntax of this operation is as follows:

```
procedure SEND_ERROR_MESSAGE(
         APPLICATION_ID    :in      TES_DEFINITIONS.T_APPLICATION_ID;
         ERROR_MESSAGE_SHORT:in     VICOS_DEFINITIONS.T_LOG_TEXT_SHORT;
         ERROR_MESSAGE_LONG :in     VICOS_DEFINITIONS.T_LOG_TEXT_LONG;
         ERROR_TYPE        :in      VICOS_DEFINITIONS.T_LOG_TYPE;
         LOCAL_TIME_TAG    :in      CGS_CALENDAR.CGS_DATE_AND_TIME
                                    := CGS_CALENDAR.NULL_DATE_AND_TIME;
         SMT_TIME_TAG      : in     CGS_CALENDAR.CGS_DATE_AND_TIME
                                    := CGS_CALENDAR.NULL_DATE_AND_TIME;
         SHOW_IT            in      BOOLEAN := TRUE;
         STATUS            :out     TES_DEFINITIONS.T_RETURN_STATUS);
```

The parameters of this operation have the following meaning:

– The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

– ERROR_MESSAGE_SHORT is a 40 character string containing a comprehensive message text. The CGS message window can be configured to always show the short text.

– ERROR_MESSAGE_LONG is a 255 character string containing a full message text. The CGS message window can be configured to always show the long text or suppress it and show it on demand only..

– ERROR_TYPE is an indication for the type of message to be sent. It is a four character string that can be used under user control. It is however recommended to use only the following values as predefined in package VICOS_DEFINITIONS:

```
UUT_LOG_GROUP              : constant string := ”UUT ”; –– msg received from UUT
INFO_MESSAGE_TYPE          : constant string := ”INFO”;
GENERAL_MESSAGE_TYPE       : constant string := ”MSG ”;
ERROR_MESSAGE_TYPE         : constant string := ”ERR ”;
WARNING_MESSAGE_TYPE       : constant string := ”WRN ”;
EXCEPTION_MESSAGE_TYPE     : constant string := ”EXC ”;
ALARM_MESSAGE_TYPE         : constant string := ”ALRM”;
```

The value ”SAS ” would also be allowed. In the test result database, this ERROR_TYPE is later on used for selecting certain messages.

– LOCAL_TIME_TAG is the CGS_CALENDAR like time tag for this message in terms of local time (wall clock time). The SAS must not provide the time–tag. In this case the API will add the current time as the time tag.

- SMT_TIME_TAG is the CGS_CALENDAR like time tag for this message in terms of SMT. The SAS must not provide the time–tag. In this case the API will add the current time as the time tag.
- SHOW_IT is a boolean flag which indicates whether or not TES will route the error message to the user interface. In any case, the message will be logged.
- STATUS is the error status returned by this call. Possible return states are:

SUCCESS	everything o.k
CURRENT_MODE_WRONG	TES is in idle or error mode or it is currently not running
COMMUNICATION_ERROR	some communication problem exists, carefully look into your system topology table, maybe TES died in the meantime
INVALID_APPLICATION_ID	The identifier is wrong, perhaps an old one isused?
OTHER_ERROR	an unexpected error occurred or the SAS has not called READ_COMMAND in time, i.e. the time–out has expired (configurable in the TES_CONFIG_FILE)

G INDEX OF ALL USER MANUAL PROCEDURES

H TEST RELATED HLCL COMMANDS

H-1 Basics

The High Level Command Language (HLCL) is the interactive counterpart of the User Control Language. (UCL). UCL acts as a pure programming language: automated procedures and libraries are edited off–line, compiled and kept in the database for later on–line execution.

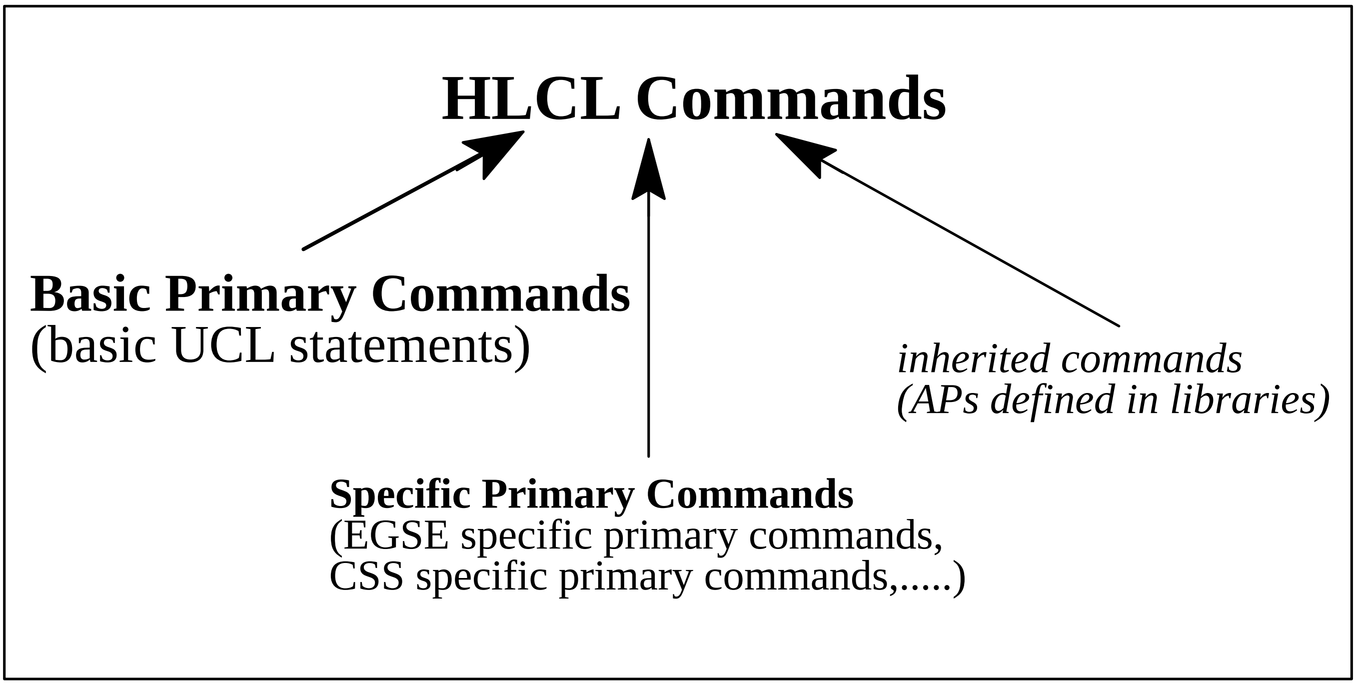


Figure 21 : *Different types of HLCL commands*

HLCL provides most of the basic language material of UCL, as well as HLCL specific extensions.

- The basic UCL statements constitute the *basic primary command* set of HLCL.
- Additionally there are the *specific primary commands*. These are commands specific to certain products, i.e. commands which will be followed by CSS only. These commands are configuration dependent, i.e. in a configuration without CSS the CSS related commands are not available.
- HLCL has the capability of dynamically extending its primary command and function set by inheriting new commands and functions from the Columbus database.

H-2 HLCL Language definition

H-2.1 Command and function classes

The following command and function classes can be distinguished:

- Import statements
  As defined in the UCL Reference Manual.
- Declarations
  As defined in the UCL Reference Manual, but HLCL does not support the declaration of procedures and functions.

- Assignments
  As defined in the UCL Reference Manual.
- Standard UCL procedures and functions (”intrinsic”)
  As defined in the UCL Reference Manual.
- Simple commands and functions
  These are predefined commands and functions. A basic set of primary commands and functions is described in this paper, but within a given application environment (EGSE, simulator, onboard) the relevant application may add application specific commands and functions. Thus the primary command set is made up of
  - basic commands and functions predefined in HLCL (see section H-7)
  - application specific commands and functions added by an application (see section H-8)
- Inherited commands and functions
  All executable objects in the Columbus database are accessible as quasi–HLCL commands or functions. They are activated in the same syntax as primary commands and functions. Inherited commands and functions comprise
  - automated procedures (APs are commands)
  - UCL library routines (are commands and functions, respectively).
- Command sequences
  Command sequences kept in the database or in files may be called like single commands (see section H-4)
- A help facility

H-2.2 Simple commands

A simple command is essentially identical to a UCL procedure call, i.e it is invoked with its name, followed by its parameter list, if any.

HLCL relaxes the UCL syntax rules to support easy typing:

- Parameter lists are given without the enclosing parenthesis.
- All parameter (open array and open string parameters) may be omitted, an appropriate default is then taken:
  - For optional parameters the default value is taken.
  - Mandatory **in** and **out** parameters are prompted for.
  - For pure **out** parameters the returned value is displayed.

H-2.3 Structured commands

The HLCL structured commands identically correspond to the UCL control structures (**if**,**case**,**loop**,**while**,**for**,**repeat**). They are only allowed in command sequences.

H-2.4 Return command

The **return** command (without a value) is only allowed in command sequences.

H-2.5 Import

The **Import** statement is identical to that in UCL.

H-2.6 Declarations and deletions

The UCL declarations are fully available in HLCL.

HLCL extends the **alias** declaration: an alias may also be assigned to a string constant. When used,the string alias must designate the name of a file that contains a command sequence. The intended effect is to be able to call a command sequence contained in a file by a simple name.

Example: `alias SHUTDOWN = ”/usr/columbus/users20/vicos/shutdown.seq”`

Note that aliases can be combined and extended in different ways:
A,B and C are parts of the system and user tree,
following combinations are allowed A/B/C, A.input, A/C.input_2

H-2.7 Assignment

Assignments are identical to UCL assignments.

H-2.8 Standard procedures and functions

All the procedures and functions defined in the UCL reference Manual may be called from HLCL, procedures becoming *commands*. HLCL predefines additional standard procedures.

H-2.9 Predefined types, literals and variables

Reserved Words:
Identical to UCL: Refer to the UCL Reference Manual

Path Identifier:
Identical to UCL: Refer to the UCL Reference Manual

Qualified Identifier:
Identical to UCL: Refer to the UCL Reference Manual

The HLCL data types are identical to the UCL types. HLCL predefines the following additional types, an application may add its own specific types.

The type OBJECTS defines literals with the following meaning as used in the LIST command:

type OBJECTS	Objects to be listed (see LIST command)
= (CONSTANTS,	all constants (including enumeration literals)
LITERALS,	all enumeration literals (subset of CONSTANTS)
VARIABLES,	all variables
TYPES,	all types
ALIASES,	all alias names
PARAMETERS	all parameters of the currently running sequence
UNITS,	all measurement units

COMMANDS	all commands (including intrinsic and procedures)
FUNCTIONS,	all functions (including intrinsic functions)
INTRINSICS,	all standard (”intrinsic”) procedures and functions
LIBRARIES,	all imported UCL libraries
SUBTREES,	the set of database subtrees authorized for the user
ITEM_TYPES,	the set of database item types authorized for the user
CALL_STACK)	the stack of loaded command sequences

There are two additional types defined for the CSS simulator.

type PULSE

A pulse is a binary event used by a simulator (CSS) function block to trigger the execution of another function block.

Objects of type PULSE are only defined in the database. They cannot be declared, have no operations and cannot be assigned. They are initially cleared and may be triggered with the standard procedure TRIGGER. Triggering a PULSE object changes its value. The value of a PULSE object may be obtained bey converting it to BOOLEAN. The conversion yields true for triggered objects, otherwise false.

type BURST_PULSE

A burst_pulse is a counting event used by a simulator (CSS) function block to trigger the execution of another function block like a pulse.

Objects of type PULSE are only defined in the database. They cannot be declared, have no operations and cannot be assigned. They are initially cleared (i.e. they have the value 0) and may be triggered with the standard procedure TRIGGER to increment their value. The default increment is 1, greater increments may be requested as a second parameter to TRIGGER. The value of a BURST_PULSE object may be obtained by converting it to UNSIGNED_INTEGER. The conversion yields a value > 0 for triggered objects, otherwise 0.

Predefined variables

HLCL predefines some variables with a special meaning. These variables have an initial default value that may be changed by the user with an assignment.

The following variables are predefined:

variable DEFAULT_PATH: **pathname** := *application defined*

Wherever a path name is prefixed with \\, the prefix is expanded with the current value of the variable DEFAULT_PATH. This variable has an initial value defined by the underlying application.

variable DEFAULT_NODE: **pathname** := *application defined*

The *default node* used to start APs and call library routines. This variable has an initial value defined by the underlying application.

variable ECHO: BOOLEAN := FALSE

This variable defines whether command executed from command sequences are echoed on the screen. If set to false, only start and stop of command sequences are reported. This variable is initially set to false.

variable TRAP: BOOLEAN := FALSE

This variable determines the error handling mechanism within command sequences. When set to true, an error in a command sequence interrupts execution of the sequence and returns control to the session. The sequence remains loaded and may be resumed with the RESUME command. When set to false, errors in a command sequence do not interrupt execution of the sequence.

variable STEP: BOOLEAN := FALSE

This variable selects *single–step execution* of command sequences. When set to true, each command from a command sequence must be confirmed before execution.

variable DEBUG: BOOLEAN := FALSE

This variable selects *debug mode*. When set to true (and a log file is open), all interactively typed commands are logged in expanded form in a commands log file (see the OPEN and CLOSE commands). When set to false, nothing is logged, even if a log file is open.

H-2.10 Units of measure

The HLCL unit of measure concept is identical to the corresponding UCL concept. The strict conformance rules of UCL are relaxed, i.e. it is possible to assign a unitized variable to a unitless value and vice versa. For more details refer to the HLCL Reference Manual.

H-2.11 Abbreviations

All identifiers may be abbreviated arbitrarily, as long as they remain unique.
For more details refer to the HLCL Reference Manual [2.3.4]

H-3 Usage

A session is the basic commanding and execution environment for interactive operation. Within a session the user types more or less the same commands in the same form that may appear in an AP in UCL.

- There is no syntactic frame that encloses a session, i.e. when logged in you are free to start typing commands.
- There is no required order or grouping of commands. Declared objects may even be deleted.

Input window

HLCL may be used in a *command window*, i.e. where a user interactively types commands or in any other environment where commands are generated by software e.g.. in the synoptic displays environment used for verification, integration and check-out.

Normal and debug mode

The user may switch between *normal mode* and *debug mode* by setting the predefined variable DEBUG to **false** or **true**, respectively.

* In normal mode commands are executed immediately.
* In debug mode each command is first displayed in expanded and evaluated form, i.e. abbreviations are removed, expressions are evaluated, defaults are inserted and all parameters are given in named notation. You are then asked for explicit confirmation before executing the command.

Command logging mode

Interactively typed commands may be collected in a log file to automatically build a *command sequence* which may then be executed again. For this purpose you can open and close command log files with the OPEN and CLOSE commands. Whenever a new file is opened, a new command sequence is built. *Command logging mode* can be controlled by the predefined variable KEEP.

* When KEEP is set to **true** (and a log file is open), all interactively entered commands are logged in the currently open log file.
* When KEEP is set to **false** (and a log file is open) commands are not logged.

Command lines

In HLCL the semi-colon is a command terminator. But the end of an input line acts as an additional command terminator. This means that practically the semi-colon may be omitted when single commands are entered, each on a separate line. It must not be omitted, however, between to commands on the same line.

Examples:

* PUT X — single command on a line, no semi-colon needed
* X := Y + Z; PUT X — two commands on a line, separated by semi-colon

H-4 Command Sequences

A command sequence is a series of single commands put in a file or in the database. They may be used to set up a certain working environment (define variables, constants, alias names etc.).

A command sequence acts as a non–interactive session. It may contain any commands, in particular it may contain calls to other command sequences. The execution of command sequences thus forms an execution hierarchy whose upper level is the session. When a command sequence is started, execution at the current level is blocked until the called sequence has completed its execution. Execution is then resumed at the point of call.

This implies that during execution of a command sequence no commands can be given from the keyboard. The execution can, however, be interrupted by pressing an interrupt key combination: the currently executed command is then cancelled, and control immediately returns to the session. The sequence is suspended but remains loaded and can be resumed at any time with the RESUME command.

Command sequence syntax

The command sequence syntax resembles that of a procedure with the following differences:

- The keyword **procedure** is replaced by the keyword **sequence**. The name given after the keyword is implicitly declared as an alias for the pathname (for sequences kept in the database) or for the file name (for sequences kept in files).
- The **begin** keyword is omitted.
- A command sequence does not have a local scope: all declarations create global objects at the session level. But the sequence parameters, as well as the alias derived from the sequence name, are visible only within the command sequence.
- Within a command sequence, a grouping of commands is not required, i.e. import statements, declarations and other commands may be mixed arbitrarily.
- Within the sequence the same rules apply as in a session, but some session specific relaxations and extensions are not valid:
  - Abbreviations are not allowed.
  - The end of a line is not a command terminator, i.e. commands may be formatted over several lines if desired but must be terminated with a semicolon (in any case).
  - Engineering units must not be omitted.

Example:

```
SEQUENCE abc;

    PUT ”HELLO WORLD”;
    ”/usr/columbus/application/vicos/test_sequences/example_2.seq”;
    PUT ”EXAMPLE_2 sequence has been executed”;

END SEQUENCE abc;
```

Echo mode

When the predefined variable ECHO is set to **true**, commands executed from a command sequence are echoed on the screen. When ECHO is not set, only start and completion of command sequences are reported.

Single step mode

Command sequences may be executed in *single step mode*. This is controlled via the predefined variable STEP.

* When STEP is set to **true** each command from a command sequence must be confirmed before execution.
* When STEP is set to **false** commands are executed automatically.

Error handling in command sequences

When an error occurs in a command sequence, the effect depends on the predefined TRAP variable. If *error trapping* is on, an error within a command in a command sequence interrupts the command sequence, as if the interrupt key combination had been pressed. Execution can then be resumed with the RESUME command.

H-5 Automated procedures (APs)

Automated procedures are programs written in UCL and compiled into an intermediate code (I_code) which is interpreted by I_Code interpreters residing in different nodes in the network.

The APs have a path name which reflects their position in the database name tree, and a parameter list. An AP can therefore be mapped on an HLCL command, the path name as the command name, and the parameter list obeying the HLCL parameter conventions. Activation of an AP can thus be expressed as an inherited quasi–HLCL command, e.g.:

\PAYLOAD\EQUIPMENT\UNIT_A\COMMAND MODE: 3

APs are executed on different nodes in a network. Several APs may be active at once, on different nodes or on the same one. The node name is obtained from the predefined variable DEFAULT_NODE.

UCL library procedures and functions are called like in UCL. The target node in the network is obtained from the predefined variable DEFAULT_NODE.
To call a library function/procedure, the function name with actual parameters is sufficient. In case of conflicts (e.g. same function name in different libraries) a qualified name is needed.
Parameters are separated as in UCL or optionally by blanks instead of ”(” resp. ”)”.
Output parameters may be assigned to declared variables or may be omitted. If omitted, the result of the parameter is printed in the command window.

H-6 Getting Help and Displaying Values

HLCL provides a help facility that allows the user to obtain a short description on an object given by its name, or on the value of an expression.

A **single question mark** displays a list of all (predefined or declared) identifiers.

The effect of a question mark applied to a (qualified) identifier depends on the identifier. If it is ambiguous, then it is assumed to be an abbreviation and a one–line information is displayed for all objects whose name matches the abbreviation. This line contains the value of the object, if any.

If the identifier is unique, a more detailed information for that object is displayed:

* For a **command or function name** a short command/function information is displayed, comprising the parameter list, and for each parameter the type(s), parameter mode and default value, if any.
* A **library** procedure/function is treated as a simple command/function.

* For an **alias**, in addition to the value of the alias an information on the designated object is displayed, depending on the type of object.
* For all **other objects** a short identification of the object is displayed, together with its value, if any.

A question mark applied to an **expression** displays the expression with all abbreviations expanded, together with its value.

For a **unit**, its definition in terms of the seven basic SI units is displayed.

The effect of a question mark applied to a database **path name** depends on the type of item:

* For a **virtual** item (non–end–item) a one–line information for each of its children is displayed.
* An **AP path** name is treated like a simple command.
* For a **library path** name a list of the library contents is displayed.

The ***recursive indicator*** **('...')** suffixed to the help command requests information to be displayed recursively:

* For a unique (qualified) identifier an information line is displayed for the identifier itself and for all identifiers referenced in its definition. This process is repeated recursively until all identifiers have been resolved.
* For a virtual database item not only its children are displayed, but also the children of the (virtual) children, and so on recursively down to end item level.

Examples:

?	– Display list of all identifiers
? ASS_PIC	– Display details of command ASSIGN_PICTURE
? ASS_PIC . . .	– Display details of ASSIGN_PICTURE recursively
? VAR1:	– Display definition and value of a variable
? MIN (INTEGER)	– Display value of an expression
? (X + Y) * Z	– Display value of an expression
? [km/h]	– Display [km/h] in terms of [m/s]
? \APM\GROUND\VICOS . . .	– Display database subtree recursively
? \APM\GROUND\VICOS . . .(UCL,HLCL)	– Display all UCL and HLCL items in database subtree

H-7 Basic primary commands

A set of primary commands is built–in to HLCL on a fixed basis. This basis comprises the different command forms: declarations, assignments, simple commands, structured commands and the help command. This section lists only the simple commands that are specific to HLCL and therefore not covered by the UCL LRM. Only the basic commands are listed here; for the different target environments additional environment specific commands may be added.

H-7.1 Delete commands

DELETE

Function: Delete (undeclare) a user declared **object** (constant, variable, type,...).

Parameters: identifier
- Mode: Mandatory
- Type: string
- Meaning: Name of the object to be deleted

Constraints: Predefined objects and objects imported from libraries cannot be deleted.

DELETE

Function: Delete a **uni**t.

Parameters: identifier
- Mode: Mandatory
- Type: unit identifier
- Meaning: Name of the object to be deleted

Constraints: The unit must be a single unit identifier in unit brackets.

Example: DELETE [km]

H-7.2 Display commands

LIST

Function: Display a list of objects of a certain class. The list may be restricted to a specific scope.

Parameters: class — The class of objects to be displayed. It is given as a value of the predefined enumeration type OBJECTS.

scope
- Mode: Optional
- Type: pathname
- Default: \\
- Meaning: This parameter allows to restrict the list of objects to a specific library scope determined by the pathname of the library. Only objects declared in that library are then listed. If no scope is given, all scopes (i.e. the session and all imported libraries) are listed.

Constraints: For database items the scope is the path name of a virtual item (non–end–item), only items within that subtree are listed.

Example: LIST VARIABLES
LIST APS, \APM\TEST_NODE_1

H-7.3 Command sequence related commands

CANCEL

Function: Remove the command sequence call stack, i.e. all currently loaded (and suspended) command sequences.

Parameters: none

Constraints: This command can only be used interactively when no command sequence is active. RESUME is then no longer possible.

CHECK

Function: Check a command sequence for syntactical correctness.

Parameters:	sequence	Mode:	Mandatory
		Type:	string or pathname
		Meaning:	Name of the command sequence (path name if in the database, string if in a file)

Constraints: Problems resulting from execution of the command sequence in different session environ–ments and from possibly violated semantical conditions between actual parameters and between commands themselves cannot be checked.

RESUME

Function: Resume command sequence execution which was suspended by the SUSPEND command, by error or user interrupt.

Parameters: none

Constraints: Not possible after the CANCEL command.

SUSPEND

Function: Suspend execution of a command sequence and return to interactive input. The sequence may later be resumed at the suspension point.

Parameters: none

Constraints: This command can only be used within a command sequence.

H-7.4 Input/output commands

GET

Function: Read a value from the keyboard, optionally write a prompt before reading. The type of the value is determined by the actual parameter.

Parameters:	prompt	Mode:	Optional
		Type:	string
		Meaning:	Write a prompt before reading.

Constraints: Invalid input will be reported as an error, and reading will be repeated until a correct value has been read.

PUT

Function: Display the value of the parameter as an output line on the screen. The parameter may be an expression of any type. (Several items may be combined into one line by converting them to string and concatenating them with the + operator.)

Parameters:	prompt	Mode:	Mandatory
		Type:	any type
		Meaning:	The item to be written

Constraints: This command does not actually write the value but only buffers it. The buffered line is output with the PUT_LINE command.

Example:

```
VARIABLE_1 : STRING := ”ALL WENT OK”
PUT  VARIABLE_1
PUT ”OK”
```

H-7.5 Command logging commands

CL0SE

Function: The currently open log file is closed, the generated command sequence is syntactically completed first.

Parameters: none

Constraints: This command is only allowed when a log file is currently open.

0PEN

Function: Opens a new command log file and generate a command sequence in it. The syntactical form of the commands in the command sequence to be generated may be specified by several parameters.

Parameters:

file	Mode:	Mandatory
	Type:	string
	Meaning:	Name of the log file to be created
sequence	Mode:	Mandatory
	Type:	string
	Meaning:	Name of the command sequence to be created in the log file
exclude	Mode:	Optional
	Type:	string
	Default:	”” (nothing is excluded)
	Meaning:	List of commands to be excluded from logging. Command names are separated by comma or blank.
named	Mode:	Optional
	Type:	boolean
	Default:	TRUE
	Meaning:	Specifies whether commands are to be generated in named notation
evaluated	Mode:	Optional
	Type:	boolean
	Default:	FALSE
	Meaning:	Specifies whether parameters are to be evaluated and given as values, or expressions are to be kept.
formatted	Mode:	Optional
	Type:	boolean
	Default:	FALSE
	Meaning:	Specifies whether commands are to be formatted with each parameter on a separate line, or unformatted with the whole command in one line.

Constraints: This command is only allowed when no log file is currently open.

H-7.6 Pulse type related commands

TRIGGER

Function: Triggers a PULSE object.

Parameters:	item	Mode:	Mandatory
		Type:	pathname
		Meaning:	Path to the desired item including the item name.

Constraints: Used only with PULSE objects.

TRIGGER

Function: Triggers a BURST_PULSE object.

Parameters:	item	Mode:	Mandatory
		Type:	pathname
		Meaning:	Path to the desired item including the item name.
	increment	Mode:	Optional
		Type:	integer
		Default:	1
		Meaning:	Determines the number of pulses emitted.

Constraints: Used only with BURST_PULSE objects.

H-8 Specific primary commands

H-8.1 EGSE Specific Primary Commands

The following list gives the list of statements that are specific user operation related commands required to control VICOS S/W

ABORT

Function: Abort an automated procedure

Parameters:

node
- Mode: Mandatory
- Type: String
- Meaning: path name of the test node where the AP is running

ap
- Mode: Mandatory
- Type: Integer
- Meaning: process number of the AP process

Example: `ABORT TEST_NODE_1, 3`

ABORT_ALL_APS

Function: Abort all automated procedure running on the addressed testnode

Example: `ABORT_ALL_APS`

ASSIGN_PICTURE

Function: Create a synoptic display and assign the picture to it

Parameters:

picture
- Mode: Mandatory
- Type: Pathname
- Meaning: Name of the picture to be displayed

width
- Mode: Optional
- Type: Integer
- Default: –1
- Meaning: Width of the synoptic window to be created

height
- Mode: Optional
- Type Integer
- Default: –1
- Meaning: Height of the synoptic window to be created

horizontal_position
- Mode: Optional
- Type Integer
- Default: –1
- Meaning: X position of the synoptic window to be created

	vertical position	Mode:	Optional
		Type	Integer
		Default:	–1
		Meaning:	Y position of the synoptic window to be created

Example:
```
ASSIGN_PICTURE \DMS_05\PICTURE07
ASS_P          \DMS_05\PICTURE07,400,400,100,0
```

GET_ENVIRONMENT

Function: Obtain value for environment name

Parameters:	name	Mode:	Mandatory
		Type:	String
		Meaning:	Name of the environment variable.
	value	Mode:	Mandatory
		Type:	String
		Meaning:	Value of the environment variable.

Example: VARIABLE SAS_HOME: STRING (100);
GET_ENVIRONMENT ”SAS_HOME”, SAS_HOME

HISTORY

Function: Display a list of previously entered commands

Parameters:	count	Mode:	Optional
		Type:	Integer
		Default:	10
		Meaning:	Number of commands to be displayed

Constraints: in command facility only

Example: `HISTORY 25`

PRINT

Function: Print a file

Parameters:	file	Mode:	Mandatory
		Type:	String
		Meaning:	File to be printed (including directory path)
	printer	Mode:	Optional
		Type:	enumeration type
		Default:	LASER_PRINTER1
		Meaning:	Name of the printer which prints the document

Example: `PRINT ”vicos/testfile”`

START_HCI_APPLICATION

Function: Start an application program or HCI window application on the workstation.

Parameters: application
Mode: Mandatory
Type: String
Meaning: Name of the program. If the application name has a **HCI.** prefix the HCI window application is started (see example 2). The list of HCI window applications is described in 8.3.2.3.3 Screen Setup Maintenance.

parameter
Mode: Optional
Type: String
Default: ””
Meaning: If the application needs parameters, they can be given here.

user_confirmed
Mode: Optional
Type: Boolean
Default: True
Meaning: If true, the user is ask for confirmation before starting the application.

Example 1: `START_HCI_APPLICATION ”${OPENWINHOME}/bin/xterm”`,””,false

Example 2: `START_HCI_APPLICATION “HCI.GRAPH_FACILITY -MEASUREMENT \MOTOR\CURRENT –GRAPH 1 –MINIMUM 0.0 –MAXIMUM 100.0”, “”, false`

H-8.2 CSS related specific primary commands

The following commands are to be handled by CSS. Within a combined CSS/VICOS simulator configuration the commands may be entered either within the CSS environment (through MOCS if CSS is running in stand–alone mode) or within the VICOS environment (through HCI).

Note that all CSS related commands are still in development.
This is only a preliminary description.

ACTIVATE

Function: Activate the actual definition

Parameters:	definition	Mode:	Optional
		Type:	enumeration type (MONITORING,LOGGING)
		Default:	LOGGING
		Meaning:	identifies the type of activation

Constraints: Requires log privilege, if definition specifier is logging.

ABORT_SIMULATION

Function: Abort a simulation, that has come into a faulty state. After aborting a simulation the user can only store a state vector or shut-down the simulator.

Parameters: none

Constraints: Requires Standalone Configuration Mode.
Requires session owner privilege.
Can be entered only in a running–faulty state of the simulator.

BROADCAST

Function: Send a message to one specified or all users connected with the same simulation.

Parameters:	message	Mode:	Mandatory
		Type:	string
		Meaning:	message contents
	user	Mode:	Optional
		Type:	string
		Default:	””
		Meaning:	specifies the user, if not set the message is send to all users

Constraints: none

*Note that the message is automatically send to **all** users. That does not dependent on wether a user name is given or not.*

CONNECT_TO_CSS_KERNEL

Function: Connect to a running CSS kernel

Parameters:	host	Mode:	Optional
		Type:	string
		Default:	the local machine, where the ICP is running
		Meaning:	name of the target machine (where CSS is running)

Constraints: a connection with a CSS UI is required

CONNECT_TO_CSS_UI

Function: Connect to a running CSS user interface.

Parameters:	host	Mode:	Optional
		Type:	string
		Default:	the local machine, where the ICP is running
		Meaning:	name of the target machine (where CSS is running)

Constraints: a connection with a CSS UI is required

DEACTIVATE

Function: Deactivate the actual definitions.

Parameters:	definition	Mode:	Optional
		Type:	enumeration type (MONITORING,LOGGING)
		Default:	LOGGING
		Meaning:	identifies the type of deactivation

Constraints: Requires log privilege, if definition specifier is logging table.

DISCONNECT_FROM_CSS_KERNEL

Function: Disconnect from a running CSS kernel, leaving it up and running

Parameters: none

Constraints: a connection with a CSS UI is required

DISCONNECT_FROM_CSS_UI

Function: Disconnect from a running CSS user interface, this is actually the same as STOP_CSS_UI.

Parameters: none

Constraints: a connection with a CSS UI is required

FREEZE

Function: The actual value of the selected model item or external input item is frozen and does not change according to the function block’s calculations.

Parameters:	item	Mode:	Mandatory
		Type:	pathname
		Meaning:	Path to the desired model item including the item name
	time	Mode:	Optional
		Type:	time format
		Default:	0.0 [s]
		Meaning:	identifies either a duration or a point in time, if it is not set the command will be performed immediately
	class	Mode:	Optional
		Type:	enumeration type (LOCAL, SMT)
		Default:	SMT
		Meaning:	gives the reference system for the time (local time or simulated mission time)

Constraints: Requires write privilege. Time parameters are optional. If they are missing the command is executed immediately.

GRANT

Function: Grant a specified privilege to a specified user on a specified host

Parameters:	privilege	Mode:	Mandatory
		Type:	enumeration type (LOG_PRIVILEGE, WRITE_PRIVILEGE, START_PRIVILEGE)
		Meaning:	the privilege log allows to log items in the model, the privilege write allows to assign values to model items, the privilege start allows to start and stop the simulation
	user	Mode:	Mandatory
		Type:	string
		Meaning:	specifies the user name
	host	Mode:	Mandatory
		Type:	string
		Meaning:	specifies the users machine

Constraints: Used only by the session owner.

LOAD

Function: A complete simulation state (state vector) is loaded by the Kernel software. A simulation state consists of concrete values for every model output and external input/output. There is always one initial (default) state generated by CTG. The user may change the state by assigning se–lected values and/or running the simulation for a controlled number of steps. The changed state can be stored as an alternative initial value set.

Parameters:	state vector	Mode:	Mandatory
		Type:	string
		Meaning:	Name of the state vector to be loaded

Constraints: Requires session owner privilege.

LOAD_TABLE

Function: Load a prepared definition set for the monitoring and logging of values, tracing of function blocks or complete definition sets including everything. One or more definition tables are made the actual definitions, that can be modified by add/remove commands.

Parameters:	table	Mode:	Mandatory
		Type:	string
		Meaning:	identifies the name of the table

Constraints: If used more than once, replaces the actual definitions.

LOG

Function: Log the specified item

Parameters:	action	Mode:	Mandatory
		Type:	enumeration type (ADD, REMOVE)
		Meaning:	describes what to do with the item
	table	Mode:	Mandatory
		Type:	string
		Meaning:	identifies the name of the table
	item	Mode:	Mandatory
		Type:	pathname
		Meaning:	Path to the desired item including the item name
	attribute	Mode:	Optional
		Type:	enumeration type (CYCLIC, ON_CHANGE)
		Default:	CYCLIC
		Meaning:	Logging mode

Constraints: Requires log privilege.

REACTIVATE

Function:	Let the value of the model item or external input item be calculated by the model function block if the simulation is active. (Switch to model item calculation – normal case)		
Parameters:	item	Mode:	Mandatory
		Type:	pathname
		Meaning:	Path to the desired model item including the item name
	time	Mode:	Optional
		Type:	time format
		Default:	0.0 [s]
		Meaning:	identifies either a duration or a point in time, if it is not set the command will be performed immediately
	class	Mode:	Optional
		Type:	enumeration type (LOCAL, SMT)
		Default:	SMT
		Meaning:	gives the reference system for the time (local time or simulated mission time)
Constraints:	Requires write privilege. Time parameters are optional. If they are missing the command is executed immediately.		

SET_ASSIGN

Function:	Set parameters for assignment of MOCS variables		
Parameters:	disable	Mode:	Optional
		Type:	enumeration type (CALCULATE, FROZEN)
		Default:	CALCULATE
		Meaning:	The value will be set to the
	time	Mode:	Optional
		Type:	time format
		Default:	0.0 [s]
		Meaning:	identifies either a duration or a point in time, if it is not set the command will be performed immediately
	class	Mode:	Optional
		Type:	enumeration type (LOCAL, SMT)
		Default:	SMT
		Meaning:	gives the reference system for the time (local time or simulated mission time)
Constraints:	Requires write privilege. Time parameters are optional. If they are missing the command is executed immediately.		

- *The user first sets the parameters with the SET_ASSIGN command. Then the value is written directly to the variable (see Examples). The parameters from the last SET_ASSIGN command (or the defaults) are valid until a new SET_ASSIGN command will be given.*

Examples:

```
SET_ASSIGN CALCULATE, 100.0, LOCAL
\APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.OUTPUT = 305.76
In 100 sec local time the value 305.76 will be written to the variable
BLOCK_A.OUTPUT.

SET_ASSIGN FROZEN
\EURECA\TEST\MODEL_1\BOOM1\VALVE3.OUTPUT = 0.0
The value will be set to zero and frozen immediately,i.e. VALVE3.OUT-
PUT has the value zero until it is released with the REACTIVATE com-
mand.
```

SET_MINFRAME

Function: Set the simulator minframe, i.e. the simulation time raster

Parameters:	duration	Mode:	Mandatory
		Type:	float
		Meaning:	determines the frame duration, i.e. how long one frame shall last in SMT

Constraints: Requires Standalone Configuration Mode.
Requires session owner privilege.
Only possible directly after the LOAD command.

- *Note that the frame duration has no influance on the simulation execution time.*

SET_SIMULATION_ATTRIBUTES

Function: Select the simulation attributes. Simulation attributes are:
activation mode (step, continuous) and
time mode (automatic and virtual).
If the simulation is started in a VICOS configuration mode/HW environment, the time mode is automatically set to realtime and the activation mode to continuous. This command must not be used then.

Parameters:	activation mode	Mode:	Mandatory
		Type:	enumeration type (STEPWISE, CONTINUOUS)
		Meaning:	mode of model execution
	steps	Mode:	Mandatory
		Type:	unsigned integer
		Meaning:	Number of steps to be performed

	time mode	Mode:	Mandatory
		Type:	enumeration type (VIRTUAL, AUTO)
		Meaning:	mode of model execution
	factor	Mode:	Optional
		Type:	real
		Default:	1.0
		Meaning:	multiples of miniframes used for one virtual frame (only for virtual time mode)

Constraints: Requires that MOCS has not been started from VICOS (i.e. CSS in stand–alone mode)

SET_TIME

Function: Set the initial mission time (SMT).

Parameters:	time	Mode:	Mandatory
		Type:	time_type

Constraints: Requires Standalone Configuration Mode.
Requires session owner privilege.

Example:
```
SET_TIME 24.12.1996 13:27:55.50
```

SNAPSHOT

Function: Write a model item value once into the logfile or into the user interaction pane.

Parameters:	item	Mode:	Mandatory
		Type:	pathname
		Meaning:	Path to the desired item including the item name
	destination	Mode:	Optional
		Type:	enumeration type (LOGFILE,SCREEN)
		Default:	SCREEN
		Meaning:	Where the value is displayed/stored.

Constraints: Log privilege is required if the destination is the log file.

Example:
```
SNAPSHOT  \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.INPUT LOGFILE
```

START_CSS_KERNEL

Function: Start the CSS kernel

Parameters: host
Mode: Mandatory
Type: string
Meaning: name of the target machine (where to start the CSS UI)

Constraints: a connection with a CSS UI is required

Example: `START_CSS_KERNEL ”css2_2”`

START_CSS_UI

Function: Start the CSS user interface

Parameters: model
Mode: Mandatory
Type: pathname
Meaning: pathname including the model name

host
Mode: Optional
Type: string
Default: the name of the local machine, where the ICP is running
Meaning: name of the target machine (where to start the CSS UI)

Constraints: none

Example: `START_CSS_UI \EURECA\SIMULATOR\TEST\MODEL_1,”css2_2”`

START_SIMULATION

Function: Start a simulation for the first time or continue a simulation after having it suspended. The number of simulation steps executed depends on the simulation mode. A defined number of steps is executed, if step mode is preselected. An unlimited number of steps is executed, if the continuous mode is preselected. The simulation can be stopped then only by a user action. In SIVQ/VICOS configuration mode the simulation is controlled by controlling simulated mission time by VICOS commands.

Parameters: none

Constraints: Requires Standalone Configuration Mode
Requires session owner privilege.

STOP_CSS_KERNEL

Function: Stop the CSS kernel

Parameters: none

Constraints: none

STOP_CSS_UI

Function: Stop the CSS user interface

Parameters: none

Constraints: none

STOP_SIMULATION

Function: Suspend a running simulation. Simulated mission time is halted, too.

Parameters: none

Constraints: Requires Standalone Configuration Mode
Requires session owner privilege.

STORE

Function: Store a complete simulation state (state vector) as an initial value set.

Parameters:	state vector	Mode:	Mandatory
		Type:	string
		Meaning:	Name of the state vector. The simulation state will be stored under this name.

Constraints: Requires session owner privilege.

TRACE

Function: Trace the execution of the specified function block. A message containing the time of the activation appears either on screen or is stored in a logfile.

Parameters:	action	Mode:	Mandatory
		Type:	enumeration type (ADD, REMOVE)
		Meaning:	describes what to do with the item
	table	Mode:	Mandatory
		Type:	string
		Meaning:	identifies the name of the table
	block_path	Mode:	Mandatory
		Type:	pathname
		Meaning:	Path to the desired function block
	destination	Mode:	Optional
		Type:	enumeration type (LOGFILE,SCREEN)
		Default:	SCREEN
		Meaning:	Message destination.

Constraints: Requires log privilege, if log file is defined as destination.

UNMONITOR

Function:	Stop the monitoring of a single item.		
Parameters:	table	Mode:	Mandatory
		Type:	string
		Meaning:	identifies the name of the table
	item	Mode:	Mandatory
		Type:	pathname
		Meaning:	Path to the desired item including the item name
Constraints:	none		

WAIT

Function:	Suspend command execution for/until. Wait for a specified time delay or up to a specified time before processing the next command. The time may be expressed as a delay (relative local time value), a local time or Simulated Mission Time.		
Parameters:	time	Mode:	Mandatory
		Type:	time type
		Meaning:	wait up to a specified time or for a specified time delay
	class	Mode:	Optional
		Type:	enumeration type (LOCAL,SMT)
		Default:	LOCAL
		Meaning:	gives the reference system for the time (local time or simulated mission time)
Constraints:	Used only in command sequences		

Example:

```
WAIT 23.08.1995 07:23:17.000
WAIT 100.0, SMT
```

WITHDRAW

Function:	Withdraw a specified privilege previously granted to a specified user.		
Parameters:	privilege	Mode:	Mandatory
		Type:	enumeration type (LOG_PRIVILEGE, WRITE_PRIVILEGE, START_PRIVILEGE)
		Meaning:	the privilege will be removed
	user	Mode:	Mandatory
		Type:	string
		Meaning:	specifies the user name
	host	Mode:	Mandatory
		Type:	string
		Meaning:	specifies the users machine

Constraints: Used only by the session owner.

The following commands are already described in H-7.6, but they are only used in connection with the CSS features.

TRIGGER

Function: Triggers a PULSE object.

Parameters:	item	Mode:	Mandatory
		Type:	pathname
		Meaning:	Path to the desired item including the item name.

Constraints: Used only with PULSE objects.

TRIGGER

Function: Triggers a BURST_PULSE object.

Parameters:	item	Mode:	Mandatory
		Type:	pathname
		Meaning:	Path to the desired item including the item name.
	increment	Mode:	Optional
		Type:	integer
		Default:	1
		Meaning:	Determines the number of pulses emitted.

Constraints: Used only with BURST_PULSE objects.

H-9 UCL System Libraries

The UCL System Libraries contain high–level real–time procedures for monitoring, control and special test purposes. They are mentioned here to give a better impression of the test and operations capabilities because – as mentioned earlier – all UCL functions are available on HLCL level, too (note that you have to use the IMPORT command first to specify and load the desired library). For detailed information about the UCL libraries refer to Appendix I.

H-9.1 System Libraries

HLCL allows to inherit complete libraries of UCL procedures and functions. UCL provides an Onboard System Library and a Ground System Library. The library functions and procedures can be called from within HLCL command window or HLCL sequences after an import of the resp. library has been done:

* IMPORT <library pathname>

For a list of procedures/functions defined in the library refer to UCL Reference Manual.

I UCL SYSTEM LIBRARIES

I-1 UCL Ground Library

The GROUND_LIBRARY contains all standard ground commands used for control of the ground system.

Library Id (Body_Id): 2

I-1.1 Routines Summary

CLOCK
CLOSE_ARCHIVE
DELAY
DISABLE_ARCHIVING
DISABLE_CONDITIONS
DISABLE_ENDITEM
DISABLE_EVL
DISABLE_MONITORING
DISPLAY_PICTURE
DOWNLOAD

ENABLE_ARCHIVING
ENABLE_CONDITIONS
ENABLE_ENDITEM
ENABLE_EVL
ENABLE_MONITORING
ENABLE_ON_BYTE_STREAM
ENABLE_ON_FLOAT
ENABLE_ON_INTEGER
ENABLE_ON_STATECODE
EXECUTE_AP

GET_ACQUISITION_STATUS
GET_AP_ID
GET_AP_STATUS
GET_APPLICATION_ID
GET_APPLICATION_NAME
GET_APPLICATION_STATUS
GET_CONDITION
GET_CONDITION_ITEM
GET_ENDITEM_MONITOR_STATUS
GET_FULL_ENDITEM_MONITOR_STATUS
GET_PROCESSING_STATE
GET_VERIFICATION_STATUS
INIT_APPLICATION
IS_LOCAL_NODE
ISSUE
ISSUE_AND_VERIFY
LOAD_APPLICATION
LOAD_UCL
LOG
NUMBER_CONDITIONS
NUMBER_OF_CONDITION_ITEMS
OWN_AP_ID*
PATHNAME_TO_STRING
READ_MESSAGE_FROM_AP*
READ_MESSAGE_FROM_APPLICATION*
READ_MESSAGE_FROM_USER*
READ_NUMBER_FROM_USER*

REMOVE_PICTURE
RESET_APPLICATION
RESUME_AP
ROUTE_TO_SAS
SEND_SIMULATED_ADU
SET_BITS_IN_SIMULATED_ADU
SET_CCSDS_APID
SET_DELTA_LIMIT
SET_DEFAULT_WORKSTATION
SET_EXCEPTION_COUNT
SET_EXPECTED_STATE
SET_EXPECTED_VALUE
SET_HIGH_LIMIT
SET_INTEGER_DELTA_LIMIT
SET_INTEGER_HIGH_LIMIT
SET_INTEGER_LOW_LIMIT
SET_LIMIT_SET
SET_LIMIT_SET_ON_BYTE_STREAM
SET_LIMIT_SET_ON_FLOAT
SET_LIMIT_SET_ON_INTEGER
SET_LIMIT_SET_ON_STATECODE
SET_LOW_LIMIT
SET_PROCESSING
SET_SIMULATED_ENDITEM_VALUE
START_ACQUISITION
START_AP_ON_BYTE_STREAM
START_AP_ON_FLOAT
START_AP_ON_INTEGER
START_AP_ON_STATECODE
START_APPLICATION
START_SMT
START_SMT_WITH_OFFSET
STOP_ACQUISITION
STOP_SMT
SUSPEND_AP
SYNCHRONISE_WITH_AP

UCL_STATUS
UNLOAD_APPLICATION
USER_EVENT

WAIT_FOR_ADU
WAIT_UNTIL
WITHDRAW_CONDITION
WRITE_MESSAGE_TO_AP*
WRITE_MESSAGE_TO_APPLICATION*
WRITE_MESSAGE_TO_USER*

NOTE: The routines marked with asterisks (*) cannot be called from HLCL, neither interactively nor out of HLCL sequences.

I-1.2 UCL Ground System Library Specification

```
-----------------------------------------------------------------------
--
-- *******************************************
--  GROUND_LIBRARY UCL System Library
-- *******************************************
--
--ABSTRACT--
--     Defines Procedures to control the execution of CGS/VICOS
--     Must be compiled for ground and with Body Id = 2
--
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  :  GROUND_LIBRARY System Library
--      VERSION      : ##V##CGS_4.4.0
--      CGS CM       : "%Z%%Z%"
--*----------------------------------------------------------------------
--CONTENTS--
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
--CHANGE HISTORY
--      PIRN 8105, Issue 1/A for CGS V4.4.0:
--         Add 'options' as new parameter to DISPLAY_PICTURE
--         Add 'options' as new parameter to READ_MESSAGE_FROM_USER and
--          READ_NUMBER_FROM_USER
--         Add new parameter to WRITE_MESSAGE_TO_USER and LOG
--         Add new command: SET_DEFAULT_WORKSTATION
--         Add new command: SET_CCSDS_APID
--         Add new command: START_SMT_WITH_OFFSET
--      PIRN 8076, Issue 1/- for CGS V4.3.0:
--         Add derived values to type definitions
--         Add Command Verification Status
--         Add procedures ISSUE_AND_VERIFY,GET_VERIFICATION_STATUS,ENABLE_CONDITION,
--         DISABLE_CONDITION
--         Allow virtual trees for set_limit_set operations
--         Update acquisition status
--      PIRN 8003, Issue 1/C for CGS V4.2.0.9: Add new procedures
--         Add new procedures GET_ACQUISITION_STATUS, GET_values
--         GET_APPLICATION_ID, GET_APPLICATION_NAME
--         Add CDU to all Virtual Pathname definitions
--         Add constants/array for UCL Status Text
--         Add conversion routines for pathnames to text / statecode to text
--      PIRN 8003, Issue 1/B for CGS V4.1.1.9:
--         update types and operations for the control of conditions
--         add UCL return constant NOT_OK
--      PIRN 8003, Issue 1/A for CGS V4.1.1.9:
--         Include operation ROUTE_TO_SAS
--         Include types and operations for control of conditions
--         Add additional parameter "ADU" to commands START_ACQUISITION,
```

```
--          STOP_ACQUISITION and ENABLE_MONITORING
--       PIRN 8003, Issue 1/- for CGS V4.1.1.8:
--          Include interfaces for setting / reading integer limits
--          Provide the value of the alarm counter for all monitor end-items
--          Include operation to identify the EGSE node
--          Include operation to get the AP_ID
--          Handling of default limit set in ENABLE_MONITORING
--       PIRN 6012, Issue 1/D for CGS V4.0:
--          Include command GET_FULL_ENDITEM_MONITOR_STATUS
--          Extend ISSUE_ITEM_NAME: add EGSE_BINARY_PACKET
--          Modified AP_EXECUTION_STATE / AP_COMPLETION_STATE
--END HISTORY
--
-----------------------------------------------------------------------

library GROUND_LIBRARY;

-----------------------------------------------------------------------
-- TYPES
-----------------------------------------------------------------------

  type USER_MESSAGE = STRING (255);

  type MSG_GROUP = STRING(4);
  type MSG_TYPE  = STRING(4);

  type VALUE_STRING = STRING (255);

  type ACQUISITION_COLLECTION =
         PATHNAME (EGSE_INTEGER_MEASUREMENT,
                   EGSE_FLOAT_MEASUREMENT,
                   EGSE_DISCRETE_MEASUREMENT,
                   EGSE_BYTE_STREAM_MEASUREMENT,
                   EGSE_INTEGER_SW_VARIABLE,
                   EGSE_FLOAT_SW_VARIABLE,
                   EGSE_DISCRETE_SW_VARIABLE,
                   EGSE_BYTE_STREAM_SW_VARIABLE,
                   EGSE_INTEGER_DERIVED_VALUE,
                   EGSE_FLOAT_DERIVED_VALUE,
                   EGSE_DISCRETE_DERIVED_VALUE,
                   EGSE_STRING_DERIVED_VALUE,
                   EGSE_MONITOR_LIST,
                   ADU_DESCRIPTION,
                   CDU,
                   VIRTUAL);

  type ADU_NAME = PATHNAME (ADU_DESCRIPTION);
                          -- ADU_DESCRIPTION covers the MDB enditem types
                          --  CCSDS_ADU_DESCRIPTION, UNSTRUCTURED_ADU_DESCRIPTION and
```

```
                        --  STRUCTURED_ADU_DESCRIPTION
                        -- (used with this meaning also in other type definitions)


type ANALOG_MONITOR_ITEM_NAME =
       PATHNAME (EGSE_INTEGER_MEASUREMENT,
                 EGSE_FLOAT_MEASUREMENT,
                 EGSE_INTEGER_SW_VARIABLE,
                 EGSE_FLOAT_SW_VARIABLE,
                 EGSE_INTEGER_DERIVED_VALUE,
                 EGSE_FLOAT_DERIVED_VALUE);

type INTEGER_MONITOR_ITEM_NAME =
       PATHNAME (EGSE_INTEGER_MEASUREMENT,
                 EGSE_INTEGER_SW_VARIABLE,
                 EGSE_INTEGER_DERIVED_VALUE);

type ROUTE_SAS_ITEM_NAME =
       PATHNAME (GDU_DESCRIPTION_LIST,
                 EGSE_ANALOG_STIMULUS,
                 EGSE_DISCRETE_STIMULUS,
                 EGSE_BINARY_PACKET,
                 EGSE_PREDEFINED_TC,
                 ADU_DESCRIPTION,
                 CDU,
                VIRTUAL);

type CCSDS_ITEM_NAME =
       PATHNAME (GDU_DESCRIPTION_LIST,
                 EGSE_PREDEFINED_TC,
                 ADU_DESCRIPTION,
                 CDU,
                VIRTUAL);

 type AP_STATE =
       (AP_RUNNING, AP_SUSPENDED, -- AP still exists in the system; current state
        AP_UNKNOWN,               -- no such AP (may be terminated)
        AP_SUCCESS, AP_FAILURE, AP_ABORTED); -- result of synchronize with AP

type AP_EXECUTION_STATE =
       AP_STATE (AP_RUNNING .. AP_UNKNOWN);  -- see AP_STATE comment

type AP_COMPLETION_STATE =
       AP_STATE (AP_UNKNOWN .. AP_ABORTED);  -- see AP_STATE comment

type AP_ID = INTEGER;

type AP_ID_LIST = ARRAY (1..20) OF AP_ID;
```

```
type AP_NAME = PATHNAME (UCL_AUTOMATED_PROCEDURE);

type APPLICATION_ERROR_MESSAGE = USER_MESSAGE;

type APPLICATION_ID = INTEGER;

type APPLICATION_NAME = STRING (20);

type APPLICATION_STATE =
       (SAS_UNLOADED, SAS_RESET, SAS_INIT, SAS_RUNNING, SAS_ABORTED);

type BIT_COUNT = INTEGER (1 .. 32);

type BYTE_STREAM_MONITOR_ITEM_NAME =
       PATHNAME (EGSE_BYTE_STREAM_MEASUREMENT,
                 EGSE_BYTE_STREAM_SW_VARIABLE,
                 EGSE_STRING_DERIVED_VALUE);

type DISCRETE_MONITOR_ITEM_NAME =
       PATHNAME (EGSE_DISCRETE_MEASUREMENT,
                 EGSE_DISCRETE_SW_VARIABLE,
                 EGSE_DISCRETE_DERIVED_VALUE);

type DOWNLOAD_ITEM_NAME = PATHNAME (EGSE_SOFTWARE);

type ISSUE_ITEM_NAME =
       PATHNAME (GDU_DESCRIPTION_LIST,
                 EGSE_ANALOG_STIMULUS,
                 EGSE_DISCRETE_STIMULUS,
                 EGSE_BINARY_PACKET,
                 EGSE_PREDEFINED_TC);

type SINGLE_GDU_NAME =
       PATHNAME (EGSE_ANALOG_STIMULUS,
                 EGSE_DISCRETE_STIMULUS,
                 EGSE_BINARY_PACKET,
                 EGSE_PREDEFINED_TC);

type LIMIT_SET_NUMBER = INTEGER (0..5);

type MONITOR_COLLECTION =
       PATHNAME (EGSE_INTEGER_MEASUREMENT,
                 EGSE_FLOAT_MEASUREMENT,
                 EGSE_DISCRETE_MEASUREMENT,
                 EGSE_BYTE_STREAM_MEASUREMENT,
                 EGSE_INTEGER_SW_VARIABLE,
                 EGSE_FLOAT_SW_VARIABLE,
                 EGSE_DISCRETE_SW_VARIABLE,
                 EGSE_BYTE_STREAM_SW_VARIABLE,
```

```
                 EGSE_INTEGER_DERIVED_VALUE,
                 EGSE_FLOAT_DERIVED_VALUE,
                 EGSE_DISCRETE_DERIVED_VALUE,
                 EGSE_STRING_DERIVED_VALUE,
                 EGSE_MONITOR_LIST,
                 CDU,
                 VIRTUAL);

type MONITOR_ITEM_NAME =
       PATHNAME (EGSE_INTEGER_MEASUREMENT,
                 EGSE_FLOAT_MEASUREMENT,
                 EGSE_DISCRETE_MEASUREMENT,
                 EGSE_BYTE_STREAM_MEASUREMENT,
                 EGSE_INTEGER_SW_VARIABLE,
                 EGSE_FLOAT_SW_VARIABLE,
                 EGSE_DISCRETE_SW_VARIABLE,
                 EGSE_BYTE_STREAM_SW_VARIABLE,
                 EGSE_INTEGER_DERIVED_VALUE,
                 EGSE_FLOAT_DERIVED_VALUE,
                 EGSE_DISCRETE_DERIVED_VALUE,
                 EGSE_STRING_DERIVED_VALUE);

type MONITOR_STATUS = (MONITORING_DISABLED, IN_LIMITS, OUT_OF_LIMITS);

type EXTENDED_MONITOR_STATUS =
       (DISABLED, NOT_ACQUIRED, NOMINAL,
        INTERRUPTED,        -- for delta limits only
        NON_NOMINAL,        -- for non-numeric items & delta limits
        OUT_OF_HIGH_LIMIT,  -- for numeric absolute limits
        OUT_OF_LOW_LIMIT);  -- dto.

type ACQUISITION_STATUS =
    (VALID,
     NOT_ACQ,
     NOT_RECVD,
     NOT_MAINTAINED,
     DISAB,
     INVALID,
     STATIC);

type LIMIT_VALUE = record
  case IS_INT : BOOLEAN
    when TRUE  : I_VALUE : INTEGER;
    when FALSE : R_VALUE : REAL;
  end case;
end record;

type OPTIONAL_NUMERIC_LIMIT = record
  case DEFINED : BOOLEAN
```

```
    when FALSE :
    when TRUE  : VALUE : LIMIT_VALUE;
  end case;
end record;

type OPTIONAL_STATECODE = record
  case DEFINED : BOOLEAN
    when FALSE :
    when TRUE  : VALUE : STATECODE;
  end case;
end record;

type OPTIONAL_VALUE_STRING = record
  case DEFINED : BOOLEAN
    when FALSE :
    when TRUE  : VALUE : VALUE_STRING;
  end case;
end record;

type NUMERIC_LIMITS = record
  HIGH,
  LOW,
  DELTA : OPTIONAL_NUMERIC_LIMIT;
end record;

type MONITOR_ITEM_CLASS = (NUMERIC, DISCRETE, BYTE_STREAM);

type FULL_MONITOR_STATUS = record
  case LIMITS_DEFINED : BOOLEAN
    when FALSE :
    when TRUE  :
      CURRENT_LIMIT_SET : INTEGER;
      NOMINAL_STATUS    : EXTENDED_MONITOR_STATUS;
      case ITEM_CLASS : MONITOR_ITEM_CLASS
        when NUMERIC     : DANGER_STATUS,
                           DELTA_STATUS,
                           DELTA_DANGER_STATUS :
                             EXTENDED_MONITOR_STATUS;
                           DANGER,
                           NOMINAL: NUMERIC_LIMITS;
        when DISCRETE    : EXPECTED_STATE  : OPTIONAL_STATECODE;
        when BYTE_STREAM : EXPECTED_STRING : OPTIONAL_VALUE_STRING;
      end case;
      ALARM_COUNT : UNSIGNED_INTEGER;
  end case;
end record;

type CMD_VERIFICATION_STATUS =
           (STARTED,           –– Verification is started
```

```
                             -- (i.e. is in Activation Delay)
          IN_PROGRESS,       -- Verification is in progress
                             -- (i.e. is in Verification Timeout)
          TC_VERIF_OK,       -- Verification performed with success
          TC_VERIF_FAILED,   -- Verification performed with no success
                             -- (i.e. Verification Timeout elapsed)
          ERROR,             -- Error during Verification Timeout
          NO_VERIFICATION,   -- No Verification defined for this enditem
          ABORTED,           -- Verification is aborted due to new issue of same
                             -- command or stop of remote TES
          UNDEFINED);        -- The verifcation status is not defined
                             -- (sending aborted due to error,
                             --   no command sent yet etc)



type NODE_NAME = PATHNAME (EGSE_NODE);

type PICTURE_ID = INTEGER;

type PICTURE_NAME = PATHNAME (WDU_GROUND_SYNOPTIC_DISPLAY);


type PRIORITY = (LOW, HIGH);

type TIME_BASE = (GUARANTEED_LOCAL_TIME, LOCAL_TIME,  SMT);

type UCL_ITEM_NAME = PATHNAME (UCL_AUTOMATED_PROCEDURE,
                               UCL_USER_LIBRARY);

type UCL_RETURN = INTEGER;
type UCL_STATUS_TEXT  = STRING(120);
type UCL_STATUS_CODES = ARRAY (1..100) OF UCL_STATUS_TEXT;

type USER_MESSAGE_NAME = PATHNAME (EGSE_USER_MESSAGE);

 type CONDITION =
        (EQUAL, NOT_EQUAL, LESS, GREATER, LESS_EQUAL, GREATER_EQUAL, IN_RANGE);

 type DISCRETE_CONDITION = CONDITION (EQUAL .. NOT_EQUAL);

 type CONDITION_TYPE =
    (INTEGER_TYPE, REAL_TYPE, STATECODE_TYPE, BYTE_STREAM_TYPE);

 type CONDITION_VAL = record
    case COND_TYPE : CONDITION_TYPE
       when INTEGER_TYPE     :    INTEGER_VALUE : INTEGER;
       when REAL_TYPE        :    REAL_VALUE : REAL;
       when STATECODE_TYPE   :    STATECODE_VALUE : STATECODE;
```

```
      when BYTE_STREAM_TYPE :    BYTE_STREAM_VALUE : VALUE_STRING;
    end case;
  end record;

  type FLOAT_MONITOR_ITEM_NAME =
        PATHNAME (EGSE_FLOAT_MEASUREMENT,
                  EGSE_FLOAT_SW_VARIABLE,
                  EGSE_FLOAT_DERIVED_VALUE);

  type ON_OFF = (ON, OFF);

  type ACTION_KIND = (PROCESSING, LIMIT, START_AP);

  type ACTION_DESCRIPTION = record
    case KIND : ACTION_KIND
      when PROCESSING  :
        PROCESS_ITEM : ACQUISITION_COLLECTION;
      when LIMIT       :
        LIMIT_ITEM   : MONITOR_COLLECTION;
        LIMIT_SET    : LIMIT_SET_NUMBER;
      when START_AP    :
        AP           : AP_NAME;
    end case;
  end record;

  type CONDITION_STATE = (IS_TRUE, IS_FALSE, IS_UNKNOWN);

------------------------------------------------------------------------
-- CONSTANTS
------------------------------------------------------------------------

 constant NULL: CHARACTER := CHARACTER (0);
 constant NULL_APPLICATION_NAME : APPLICATION_NAME  := "";

  constant DEFAULT_ARCH_FILE_OPEN_PERIOD: DURATION := 1800.0 [s];

  constant DEFAULT_ISSUE_TIMEOUT: DURATION := 0.5 [s];

  constant DEFAULT_LIMIT_SET: LIMIT_SET_NUMBER := 0;

  constant DEFAULT_MESSAGE: USER_MESSAGE := "";

  constant ALL_CONDITIONS : INTEGER := 0;

  constant OK:                         UCL_RETURN :=   1;
  constant AP_IS_EXECUTING:            UCL_RETURN :=   2;
  constant INVALID_PARAMETER:          UCL_RETURN :=   3;
  constant DISC_FULL:                  UCL_RETURN :=   4;
  constant INVALID_AP_ID:              UCL_RETURN :=   5;
```

```
constant INVALID_AT_TIME:              UCL_RETURN :=   6;
constant INVALID_CYCLE:                UCL_RETURN :=   7;
constant INVALID_EXC_COUNT:            UCL_RETURN :=   8;
constant INVALID_ITEM_NAME:            UCL_RETURN :=   9;
constant INVALID_LIMIT:                UCL_RETURN :=  10;
constant INVALID_LIMIT_SET:            UCL_RETURN :=  11;
constant INVALID_APPLICATION_ID:       UCL_RETURN :=  12;
constant INVALID_NODE_NAME:            UCL_RETURN :=  13;
constant INVALID_OFFSET:               UCL_RETURN :=  14;
constant INVALID_PICTURE_ID:           UCL_RETURN :=  15;
constant INVALID_PICTURE_NAME:         UCL_RETURN :=  16;
constant INVALID_SIZE:                 UCL_RETURN :=  17;
constant ITEM_IS_DISABLED:             UCL_RETURN :=  18;
constant ITEMS_NOT_ACQUIRED:           UCL_RETURN :=  19;
constant MESSAGE_TOO_BIG:              UCL_RETURN :=  20;
constant NO_ADU_SERVICE:               UCL_RETURN :=  21;
constant NO_DELTA_LIMIT:               UCL_RETURN :=  22;
constant APPLICATION_NACK:             UCL_RETURN :=  23;
constant INVALID_APPLICATION_NAME:     UCL_RETURN :=  24;
constant APPLICATION_NOT_READY:        UCL_RETURN :=  25;
constant TIMEOUT:                      UCL_RETURN :=  26;
constant TOO_MANY_SYNOPTICS:           UCL_RETURN :=  27;
constant TOO_MANY_APS:                 UCL_RETURN :=  28;
constant URT_UNKNOWN:                  UCL_RETURN :=  29;
constant INVALID_ADU_TYPE:             UCL_RETURN :=  30;
constant INVALID_TESTNODE_MODE:        UCL_RETURN :=  31;
constant ITEM_NOT_SAS_COMPATIBLE:      UCL_RETURN :=  32;
constant ITEM_NOT_FEE_COMPATIBLE:      UCL_RETURN :=  33;
constant ITEM_NOT_UUT_COMPATIBLE:      UCL_RETURN :=  34;
constant NO_GDU_SERVICE:               UCL_RETURN :=  35;
constant ITEM_IS_ENABLED:              UCL_RETURN :=  36;
constant INVALID_TIME:                 UCL_RETURN :=  37;
constant TIME_SERVICE_FAILED:          UCL_RETURN :=  38;
constant OPERATION_ABORTED:            UCL_RETURN :=  39;
constant NOT_MTP:                      UCL_RETURN :=  40;
constant ARCHIVE_FILE_ERROR:           UCL_RETURN :=  41;
constant AP_NOT_COMPILED:              UCL_RETURN :=  42;
constant AP_IS_SUSPENDED:              UCL_RETURN :=  43;
constant ARCHIVING_IS_DISABLED:        UCL_RETURN :=  44;
constant WORKSTATION_NOT_CONNECTED:    UCL_RETURN :=  45;
constant WORKSTATION_NOT_READY:        UCL_RETURN :=  46;
constant INPUT_CANCELLED:              UCL_RETURN :=  47;
constant PROTOCOL_ERROR:               UCL_RETURN :=  48;
constant SYNOPTIC_RESOURCES_EXCEEDED:  UCL_RETURN :=  49;
constant CONNECTION_TIMEOUT:           UCL_RETURN :=  50;
constant CONNECTION_ABORTED:           UCL_RETURN :=  51;
constant APPLICATION_LOAD_FAILED:      UCL_RETURN :=  52;
constant APPLICATION_IS_CONNECTED:     UCL_RETURN :=  53;
constant COMMUNICATION_ERROR:          UCL_RETURN :=  54;
```

```
  constant INVALID_FILE_TYPE:           UCL_RETURN :=  55;
  constant INVALID_NODE_TYPE:           UCL_RETURN :=  56;
  constant NOT_OK:                      UCL_RETURN :=  57;
                                             –– for values 58–60: see GROUND_TO_OB_LIB
  constant AUTHORIZATION_FAILED:        UCL_RETURN :=  61;
  constant RUNTIME_ERROR:               UCL_RETURN := 100;

  constant MIN_APPL_ERROR_CODE:         UCL_RETURN := 1000;
  constant MAX_APPL_ERROR_CODE:         UCL_RETURN := 9999;
                         –– reserved range for applications
                         –– Note: Applications (SAS) may return
                         –– their own code to TES. This code
                         –– is given to the caller in case of
                         –– no error situation in the STATUS
                         –– parameter of each procedure (i.e. instead of OK)


–– Constants to allow for easy conversion of UCL Status code to readable text

  constant UCL_DEFAULT_TEXT: UCL_STATUS_TEXT := ”Undefined UCL Status Code”;

 constant STATUS_STRING: UCL_STATUS_CODES      := (
     ”OK”,                                                                        –– 1)
     ”AP_IS_EXECUTING/BUSY: Function could not be executed: is currently/already executing”,
–– 2)
     ”INVALID_PARAMETER: A parameter is invalid or in conflict with another parameter”, –– 3)
     ”DISC_FULL: The disc of the test node or the DB Server is full”, –– 4)
     ”INVALID_AP_ID: The ID for the Automated Procedure is invalid / The addressed AP is not
running (anymore)”, –– 5)
     ”INVALID_AT_TIME: The value given for the AT_TIME parameter is invalid”, –– 6)
     ”INVALID_CYCLE: The value given for the CYCLE parameter is invalid, range is 1 minute to
23 hours 59 minutes”, –– 7)
     ”INVALID_EXC_COUNT:The value given for the Exception Count parameter is invalid”, –– 8)
     ”INVALID_ITEM_NAME: The name for the item is invalid/not existing/of wrong type”, –– 9)
     ”INVALID_LIMIT: The value for the limit is invalid”, –– 10)
     ”INVALID_LIMIT_SET: The limit set number must be in range 0..5”, –– 11)
     ”INVALID_APPLICATION_ID: The application id is not known”, –– 12)
     ”INVALID_NODE_NAME: The name for the node is invalid”, –– 13)
     ”INVALID_OFFSET: Offset is not in valid range”, –– 14)
     ”INVALID_PICTURE_ID”, –– 15)
     ”INVALID_PICTURE_NAME”,                                                     –– 16)
     ”INVALID_SIZE”,                                                             –– 17)
     ”ITEM_IS_DISABLED: Function could not be executed, because item is disabled”, –– 18)
     ”ITEMS_NOT_ACQUIRED: Function could not be executed, because item is not acquired”, ––
19)
     ”MESSAGE_TOO_BIG: (e.g.: Parameter List for SWOPs do not fit into CCSDS packet)”, –– 20)
     ”NO_ADU_SERVICE: The addressed SAS has not announced its ADU service yet”, –– 21)
     ”NO_DELTA_LIMIT”, –– 22)
     ”APPLICATION_NACK: The SAS has given a negative acknowledge”, –– 23)
```

```
    ”INVALID_APPLICATION_NAME: SAS name not known / not related / not connected ”, –– 24)
    ”APPLICATION_NOT_READY: Something went wrong when preparing/sending a cmd/request to SAS
or when receiving the acknowlege”,–– 25)
    ”TIMEOUT: The acknowlegement of the SAS was not received in time”, –– 26)
    ”TOO_MANY_SYNOPTICS: The number of synoptics already loaded exceeds the allowable limit”,
–– 27)
    ”TOO_MANY_APS: The number of APs already loaded exceeds the allowable limit”, –– 28)
    ”URT_UNKNOWN”,  –– 29)
    ”INVALID_ADU_TYPE: The type of the ADU is not compatible with the called function or the
ADU is not simulated/acquired”,–– 30)
    ”INVALID_TESTNODE_MODE: The mode of the testnode (TES) does not allow for this function”,
             –– 31)
    ”ITEM_NOT_SAS_COMPATIBLE: Error when downloading file to SAS”,
      –– 32)
    ”ITEM_NOT_FEE_COMPATIBLE”, –– 33)
    ”ITEM_NOT_UUT_COMPATIBLE”, –– 34)
    ”NO_GDU_SERVICE:The addressed SAS has not announced its GDU service yet”, –– 35)
    ”ITEM_IS_ENABLED: Function could not be executed, because item isenabled”,
      –– 36)
    ”INVALID_TIME: Time parameter or time in CCSDS Header invalid”, –– 37)
    ”TIME_SERVICE_FAILED: The access to the time service failed”, –– 38)
    ”OPERATION_ABORTED: Operation was aborted (e.g. by forced_stop of test node or by
abort_ap)”, –– 39)
    ”NOT_MTP: The addressed test node requires to be the MTP”, –– 40)
    ”ARCHIVE_FILE_ERROR: The access to an archive file failed”, –– 41)
    ”AP_NOT_COMPILED: The addressed AP needs to be compiled in the MDB”, –– 42)
    ”AP_IS_SUSPENDED: Function could not be executed, because AP is suspended”, –– 43)
    ”ARCHIVING_IS_DISABLED: Function could not be executed, because archiving is (already)
disabled”, –– 44)
    ”WORKSTATION_NOT_CONNECTED: Addressed workstation (HCI) is not connected to the test
node”, –– 45)
    ”WORKSTATION_NOT_READY:Addressed workstation (HCI) is not ready”, –– 46)
    ”INPUT_CANCELLED”, –– 47)
    ”PROTOCOL_ERROR: Something went wrong when communicating with other software parts / In-
valid Response Packet for SWOP”, –– 48)
    ”SYNOPTIC_RESOURCES_EXCEEDED or PARAMETER_ERROR: Parameter List of SWOP has wrong data /
wrong types”, –– 49)
    ”CONNECTION_TIMEOUT”, –– 50)
    ”CONNECTION_ABORTED”, –– 51)
    ”APPLICATION_LOAD_FAILED: The loading of the SAS executable failed”, –– 52)
    ”APPLICATION_IS_CONNECTED: The SAS is already connected”, –– 53)
    ”COMMUNICATION_ERROR: Error on the communication channel”, –– 54)
    ”INVALID_FILE_TYPE”, –– 55)
    ”INVALID_NODE_TYPE”, –– 56)
    ”NOT_OK: For Operations on Lists: Some of the items could not be processed / executed”,
–– 57)
    ”SW_COMMAND_NOT_FOUND: The addressed Software Command was not found”, –– 58)
    ”RESPONSE_TOO_SHORT: The response packet for a Software Command did not contain all ex-
pected parameters”, –– 59)
```

```
     ”RESPONSE_TIMEOUT: The response packet for a Software Command was not received in time”,
-- 60)
     ”AUTHORIZATION_FAILED: A critical command (GDU) was not authorized”,-- 61)
    UCL_Default_Text,UCL_Default_Text,UCL_Default_Text, -- 62..64
    UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,--
65..99
    UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,--
70..74
    UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,--
75..79
    UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,--
80..84
    UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,--
85..89
    UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,--
90..94
    UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,--
95..99
     ”RUNTIME_ERROR: Something went wrong in the software”); -- 100);



------------------------------------------------------------------------
-- PROCEDURES & FUNCTIONS
------------------------------------------------------------------------

------------------------------------------------------------------------
-- Monitoring
------------------------------------------------------------------------

procedure ENABLE_MONITORING
            (in  ITEM: MONITOR_COLLECTION;
             in  LIMIT_SET: LIMIT_SET_NUMBER := DEFAULT_LIMIT_SET;
             in  ADU: ADU_NAME := \\;
             out STATUS: UCL_RETURN);  -- #1

procedure DISABLE_MONITORING
            (in  ITEM: MONITOR_COLLECTION;
             out STATUS: UCL_RETURN);  -- #2

procedure SET_HIGH_LIMIT
            (in  ITEM: ANALOG_MONITOR_ITEM_NAME;
             in  LIMIT: REAL;
             out STATUS: UCL_RETURN);  -- #3

procedure SET_INTEGER_HIGH_LIMIT
            (in  ITEM: INTEGER_MONITOR_ITEM_NAME;
             in  LIMIT: INTEGER;
             out STATUS: UCL_RETURN);  -- #4
```

```
procedure SET_LOW_LIMIT
            (in  ITEM: ANALOG_MONITOR_ITEM_NAME;
             in  LIMIT: REAL;
             out STATUS: UCL_RETURN);  –– #5

procedure SET_INTEGER_LOW_LIMIT
            (in  ITEM: INTEGER_MONITOR_ITEM_NAME;
             in  LIMIT: INTEGER;
             out STATUS: UCL_RETURN);  –– #6

procedure SET_DELTA_LIMIT
            (in  ITEM: ANALOG_MONITOR_ITEM_NAME;
             in  LIMIT: REAL;
             out STATUS: UCL_RETURN);  –– #7

procedure SET_INTEGER_DELTA_LIMIT
            (in  ITEM: INTEGER_MONITOR_ITEM_NAME;
             in  LIMIT: INTEGER;
             out STATUS: UCL_RETURN);  –– #8

procedure SET_EXCEPTION_COUNT
            (in  ITEM: MONITOR_ITEM_NAME;
             in  N_COUNT: INTEGER;
             out STATUS: UCL_RETURN);  –– #9

procedure SET_EXPECTED_STATE
            (in  ITEM: DISCRETE_MONITOR_ITEM_NAME;
             in  STATE: STATECODE;
             out STATUS: UCL_RETURN);  –– #10

procedure SET_EXPECTED_VALUE
            (in  ITEM: BYTE_STREAM_MONITOR_ITEM_NAME;
             in  VALUE: STRING;
             out STATUS: UCL_RETURN);  –– #11

procedure SET_LIMIT_SET
            (in  ITEM: MONITOR_COLLECTION;
             in  LIMIT_SET: LIMIT_SET_NUMBER;
             out STATUS: UCL_RETURN);  –– #12

procedure GET_ENDITEM_MONITOR_STATUS
            (in  ITEM: MONITOR_ITEM_NAME;
             out MONITORING_STATUS: MONITOR_STATUS;
             out STATUS: UCL_RETURN);  –– #13

procedure GET_FULL_ENDITEM_MONITOR_STATUS
            (in  ITEM: MONITOR_ITEM_NAME;
             out MONITORING_STATUS: FULL_MONITOR_STATUS;
```

```
             out STATUS: UCL_RETURN);  -- #14

procedure GET_ACQUISITION_STATUS
            (in  ITEM: MONITOR_ITEM_NAME;
             out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
             out STATUS: UCL_RETURN);  -- #15

procedure GET_INTEGER
            (in  ITEM: INTEGER_MONITOR_ITEM_NAME;
             out VALUE: INTEGER;
             out TIME_TAG: TIME;
             out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
             out STATUS: UCL_RETURN);  -- #16

procedure GET_FLOAT
            (in  ITEM: FLOAT_MONITOR_ITEM_NAME;
             out VALUE: REAL;
             out TIME_TAG: TIME;
             out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
             out STATUS: UCL_RETURN);  -- #17

procedure GET_STATECODE
            (in  ITEM: DISCRETE_MONITOR_ITEM_NAME;
             out VALUE: STATECODE;
             out TIME_TAG: TIME;
             out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
             out STATUS: UCL_RETURN);  -- #18

procedure GET_BYTE_STREAM
            (in  ITEM: BYTE_STREAM_MONITOR_ITEM_NAME;
             out VALUE: STRING;
             out TIME_TAG: TIME;
             out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
             out STATUS: UCL_RETURN);  -- #19
-----------------------------------------------------------------------
-- Time Management
-----------------------------------------------------------------------

function  CLOCK (in BASE: TIME_BASE := LOCAL_TIME) : TIME;  -- #20

procedure DELAY
            (in  DELAY: DURATION;
             in  BASE: TIME_BASE := LOCAL_TIME);  -- #21

procedure WAIT_UNTIL
            (in  DUE_TIME: TIME;
             in  BASE: TIME_BASE := LOCAL_TIME;
             out STATUS: UCL_RETURN);  -- #22
```

```
procedure START_SMT
            (in  VALUE: TIME := ~:~ ;
             out STATUS: UCL_RETURN);  –– #23

procedure START_SMT_WITH_OFFSET
            (in  OFFSET: INTEGER;
             out STATUS: UCL_RETURN);  –– #24

procedure STOP_SMT (out STATUS: UCL_RETURN);  –– #25


––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
–– AP Handling
––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

procedure EXECUTE_AP
            (in  AP: AP_NAME ();
             in  PRIO: PRIORITY := LOW;
             in  NODE: NODE_NAME := \\;
             out ID: AP_ID;
             out STATUS: UCL_RETURN);  –– #26

procedure SYNCHRONISE_WITH_AP
            (in  AP: AP_ID;
             in  NODE: NODE_NAME :=  \\;
             out COMPLETION: AP_COMPLETION_STATE;
             out STATUS: UCL_RETURN);  –– #27

procedure SUSPEND_AP
            (in  AP: AP_ID;
             in  NODE: NODE_NAME := \\;
             out STATUS: UCL_RETURN);  –– #28


procedure RESUME_AP
            (in  AP: AP_ID;
             in  NODE: NODE_NAME := \\;
             out STATUS: UCL_RETURN);  –– #29

function OWN_AP_ID : AP_ID;  –– #30

procedure GET_AP_ID
            (in  AP: AP_NAME;
             in  NODE: NODE_NAME := \\;
             out NUMBER: INTEGER;
             out ID_LIST: AP_ID_LIST;
             out STATUS: UCL_RETURN);  –– #31

procedure GET_AP_STATUS
```

```
            (in  AP: AP_ID;
             in  NODE: NODE_NAME := \\;
             out EXEC_STATE: AP_EXECUTION_STATE;
             out STATUS: UCL_RETURN);  -- #32

procedure READ_MESSAGE_FROM_AP
            (in  BLOCK: BOOLEAN := TRUE;
             out AP: AP_ID;
             out NODE_NAME: USER_MESSAGE;
             out MESSAGE : USER_MESSAGE;
             out STATUS: UCL_RETURN);  -- #33

procedure WRITE_MESSAGE_TO_AP
            (in  AP: AP_ID;
             in  NODE: NODE_NAME := \\;
             in  MESSAGE: USER_MESSAGE := DEFAULT_MESSAGE;
             out STATUS: UCL_RETURN);  -- #34


-----------------------------------------------------------------------
-- Application Handling
-----------------------------------------------------------------------

procedure LOAD_APPLICATION
            (in  NAME: APPLICATION_NAME;
             in  NODE: NODE_NAME := \\;
             in  PARAMS: USER_MESSAGE := DEFAULT_MESSAGE;
             out APPL: APPLICATION_ID;
             out  STATUS: UCL_RETURN);  -- #35

procedure UNLOAD_APPLICATION
            (in  APPL: APPLICATION_ID;
             out STATUS: UCL_RETURN);  -- #36

procedure INIT_APPLICATION
            (in  APPL: APPLICATION_ID;
             in  MESSAGE: USER_MESSAGE := DEFAULT_MESSAGE;
             out  STATUS: UCL_RETURN);  -- #37

procedure START_APPLICATION
            (in  APPL: APPLICATION_ID;
             out STATUS: UCL_RETURN);  -- #38

procedure RESET_APPLICATION
            (in  APPL: APPLICATION_ID;
             out STATUS: UCL_RETURN);  -- #39

procedure GET_APPLICATION_STATUS
            (in  APPL: APPLICATION_ID;
```

```
             out STATE: APPLICATION_STATE;
             out ERROR_COUNT: INTEGER;
             out ERROR_MESSAGE: APPLICATION_ERROR_MESSAGE;
             out STATUS: UCL_RETURN);  –– #40

procedure READ_MESSAGE_FROM_APPLICATION
            (in  BLOCK: BOOLEAN := TRUE;
             out APPL: APPLICATION_ID;
             out MESSAGE: USER_MESSAGE;
             out STATUS: UCL_RETURN);  –– #41

procedure WRITE_MESSAGE_TO_APPLICATION
            (in  APPL: APPLICATION_ID;
             in  MESSAGE: USER_MESSAGE;
             out STATUS: UCL_RETURN);  –– #42

procedure GET_APPLICATION_NAME
            (in  APPL: APPLICATION_ID;
             out NAME: APPLICATION_NAME;
             out STATUS: UCL_RETURN);  –– #43

procedure GET_APPLICATION_ID
            (in  NAME: APPLICATION_NAME;
             out APPL: APPLICATION_ID;
             out STATUS: UCL_RETURN);  –– #44
––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
–– Issue of HW Stimuli & Downloading
––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

procedure ISSUE
            (in  ITEM: ISSUE_ITEM_NAME();
             in  PRIO: PRIORITY := LOW;
             in  AT_TIME: TIME := ~:~;
             in  BASE: TIME_BASE := LOCAL_TIME;
             in  ONBOARD_EXECUTION_TIME: TIME := ~:~;
             in  TIMEOUT: DURATION := default_issue_timeout;
             out STATUS: UCL_RETURN);                           –– #45

procedure ISSUE_AND_VERIFY
       (in ITEM: SINGLE_GDU_NAME();
        in PRIO: PRIORITY := LOW;
        in AT_TIME: TIME := ~:~;
        in BASE: TIME_BASE := LOCAL_TIME;
        in ONBOARD_EXECUTION_TIME: TIME := ~:~;
        in TIMEOUT: DURATION := default_issue_timeout;
        in ACTIVATION_DELAY: DURATION := –1.0 [s];
        in VERIFICATION_TIMEOUT : DURATION := –1.0 [s];
        in WAIT_TIL_END: BOOLEAN := FALSE;
        out VERIFICATION_STATUS: CMD_VERIFICATION_STATUS;
```

```
       out STATUS: UCL_RETURN);                              -- #46

procedure GET_VERIFICATION_STATUS
       (in ITEM: SINGLE_GDU_NAME;
        out VERIFICATION_STATUS: CMD_VERIFICATION_STATUS;
        out STATUS: UCL_RETURN);                             -- #47

procedure ENABLE_ENDITEM
            (in  ENDITEM: ISSUE_ITEM_NAME;
             out STATUS: UCL_RETURN);                          -- #48

procedure DISABLE_ENDITEM
            (in  ENDITEM: ISSUE_ITEM_NAME;
             out STATUS: UCL_RETURN);                          -- #49

procedure DOWNLOAD
            (in  APPL: APPLICATION_ID;
             in  ITEM: DOWNLOAD_ITEM_NAME;
             in  NODE: NODE_NAME := \\;
             out STATUS: UCL_RETURN);                          -- #50


-----------------------------------------------------------------------
-- Raw Data Handling
-----------------------------------------------------------------------

procedure START_ACQUISITION
            (in  ITEM: ACQUISITION_COLLECTION;
                      in  ADU: ADU_NAME := \\;
             out STATUS: UCL_RETURN);  -- #51

procedure STOP_ACQUISITION
            (in  ITEM: ACQUISITION_COLLECTION;
                      in  ADU: ADU_NAME := \\;
             out STATUS: UCL_RETURN);  -- #52

procedure SEND_SIMULATED_ADU
            (in  ADU: ADU_NAME;
             out STATUS: UCL_RETURN);  -- #53

procedure WAIT_FOR_ADU
            (in  ADU: ADU_NAME;
             out TIME_TAG: TIME;
             out SEQUENCE_NUMBER: INTEGER;
             out STATUS: UCL_RETURN);  -- #54

procedure SET_BITS_IN_SIMULATED_ADU
            (in  ADU: ADU_NAME;
             in  OFFSET: INTEGER;
```

```
            in  SIZE: BIT_COUNT;
            in  VALUE: BITSET;
            out STATUS: UCL_RETURN);  -- #55

procedure SET_SIMULATED_ENDITEM_VALUE
           (in  ITEM: MONITOR_ITEM_NAME;
            in  BOOL_VALUE: BOOLEAN :=  FALSE;
            in  INT_VALUE: INTEGER :=  0;
            in  BITSET_VALUE: BITSET :=  {};
            in  REAL_VALUE: REAL :=  0.0;
            in  STR_VALUE: USER_MESSAGE :=  "";
            out STATUS: UCL_RETURN);  -- #56

procedure ROUTE_TO_SAS
           (in ITEM: ROUTE_SAS_ITEM_NAME ;
            in SAS_NAME: APPLICATION_NAME;
            in OLD_SAS_NAME: APPLICATION_NAME :=  NULL_APPLICATION_NAME;
            out STATUS: UCL_RETURN);  -- #57

procedure SET_CCSDS_APID
           (in ITEM: CCSDS_ITEM_NAME ;
            in CCSDS_APID: INTEGER;
            in OLD_CCSDS_APID: INTEGER := -1;
            out STATUS: UCL_RETURN);  -- #58
-----------------------------------------------------------------------
-- Event Handling
-----------------------------------------------------------------------

procedure LOG (in  MESSAGE: USER_MESSAGE;
                 in  LONG_TEXT: USER_MESSAGE := "";
              out STATUS: UCL_RETURN);  -- #59

procedure USER_EVENT
           (in  MESSAGE: USER_MESSAGE;
            out STATUS: UCL_RETURN);  -- #60


-----------------------------------------------------------------------
-- Engineering Value Logging
-----------------------------------------------------------------------

procedure ENABLE_EVL
           (in  ITEM: MONITOR_COLLECTION;
            out STATUS: UCL_RETURN);  -- #61

procedure DISABLE_EVL
           (in ITEM: MONITOR_COLLECTION;
            out STATUS: UCL_RETURN);  -- #62
```

```
-----------------------------------------------------------------------
-- Archiving
-----------------------------------------------------------------------

procedure ENABLE_ARCHIVING
            (in  CYCLE: DURATION := DEFAULT_ARCH_FILE_OPEN_PERIOD;
             out STATUS: UCL_RETURN);  -- #63

procedure DISABLE_ARCHIVING;  -- #64

procedure CLOSE_ARCHIVE
            (out CLOSE_TIME : TIME;
             out STATUS: UCL_RETURN);  -- #65


-----------------------------------------------------------------------
-- User Input & Output
-----------------------------------------------------------------------

procedure READ_MESSAGE_FROM_USER
            (in  PROMPT: USER_MESSAGE;
             in  WORKSTATION: NODE_NAME := \\;
             in  OPTIONS: VALUE_STRING := "";
             out USER_ENTRY: USER_MESSAGE;
             out STATUS: UCL_RETURN);  -- #66

procedure WRITE_MESSAGE_TO_USER
            (in  MESSAGE: USER_MESSAGE;
                in  SUPPLEMENT: USER_MESSAGE := "";
             in  MESSAGE_GROUP: MSG_GROUP := "";
             in  MESSAGE_TYPE: MSG_TYPE := "MSG ";
             in  WORKSTATION: NODE_NAME :=\\;
             in  ALL_WORKSTATIONS : BOOLEAN := FALSE;
             out STATUS: UCL_RETURN);  -- #67

procedure READ_NUMBER_FROM_USER
            (in  PROMPT_MESSAGE: USER_MESSAGE;
             in  WORKSTATION: NODE_NAME := \\;
             in  OPTIONS: VALUE_STRING := "";
             out USER_ENTRY: REAL;
             out STATUS: UCL_RETURN);  -- #68


-----------------------------------------------------------------------
-- Synoptic Display Control
-----------------------------------------------------------------------
```

```
procedure DISPLAY_PICTURE
            (in  PICTURE: PICTURE_NAME;
             in  WORKSTATION: NODE_NAME := \\;
             in  OPTIONS: VALUE_STRING := ””;
             out ID: PICTURE_ID;
             out STATUS: UCL_RETURN);  –– #69

procedure REMOVE_PICTURE
            (in  ID: PICTURE_ID;
             in  WORKSTATION: NODE_NAME := \\;
             out STATUS: UCL_RETURN);  –– #70


–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
–– UCL Code Reloading from MDB
–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

procedure LOAD_UCL
            (in  ITEM: UCL_ITEM_NAME;
             out STATUS: UCL_RETURN);  –– #71


–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
–– Test Node management
–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

function IS_LOCAL_NODE (in NODE : NODE_NAME) : BOOLEAN;        –– #72

procedure SET_DEFAULT_WORKSTATION
            (in  WORKSTATION: NODE_NAME :=\\;
             out STATUS: UCL_RETURN);                          –– #73
–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
–– Conditions
–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

   ––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
   –– Concept :
   –– –––––––––
   ––
   –– The conditional monitoring will be defined by linking 1 measurement or
   –– software variable to one or more end items.
   ––
   –– Every link can be described by
   ––
   ––   ”when value of CONDITION_ITEM <MATCHES> a CONDITION_VALUE”
   ––   ”then <PERFORM_AN_ACTION> on an ITEM”
   ––
   –– where <MATCHES> can be
   ––    EQUAL, NOT_EQUAL, LESS, GREATER, LESS OR EQUAL, GREATER OR EQUAL
```

```
--
--    (Note: the IN_RANGE alternative is currently not supported. It is included
--     in the definition of the CONDITION type only for MDB compatibility reasons )
--
-- where <PERFORM_AN_ACTION> can be
--
--    "enable the processing of a measurement / a sw_variable / the
--                                measurements and sw variables of a
--                                monitoring list / the measurements of an
--                                ADU / the measurements and sw variables
--                                of a subtree"
--    "set a new limit set for a measurement / a sw_variable"
--    "start an AP"
--
-- When processing CONDITION_ITEM, all conditions linked to that item will
-- be analysed and for those that are true, the defined action will be
-- performed.
-- It is necessary to start the acquisition of CONDITION_ITEM to allow the
-- condition to be analysed.
--
-- For performance reasons, the conditions will only be analysed when the
-- value of the CONDITION_ITEM is modified, and not everytime the
-- CONDITION_ITEM is processed.
-- However, the when defining a condition, it will be checked immediately
-- if the CONDITION_ITEM has a valid value.
--
-- Note that the order of processing of the data from one ADU is the order of
-- the definition of the items in the ADU.
-- So if an adu contains an ITEM and its CONDITION_ITEM, to ensure that the
-- CONDITION_ITEM is processed before the ITEM, it must be defined in the
-- list of items of the ADU description before the ITEM.
--------------------------------------------------------------------



--------------------------------------------------------------------
-- ENABLE/DISABLE  CONDITION
--
--                        ------------------------
-- Will enable/disable a condition or all conditions of
--  measurement(s), sw variable(s) and derived value(s).

-- PARAMETER DESCRIPTION :
-- inputs
-- ------
-- ITEM : measurement(s), derived values(s) or software variable(s) carrying the
--                  condition.
--
-- outputs
```

```
   -- -------
   -- STATUS : the return status of the operation
   --    OK                 : operation successfull
   --    INVALID_TESTNODE_MODE : the test node is not in executing mode
   --    INVALID_ITEM_NAME : ITEM is not managed locally
   --    RUNTIME_ERROR     : an unexpected error occurred

procedure ENABLE_CONDITIONS
              (in  ITEM: ACQUISITION_COLLECTION;
               out STATUS: UCL_RETURN);                     -- #74

procedure DISABLE_CONDITIONS
              (in  ITEM: ACQUISITION_COLLECTION;
              out STATUS: UCL_RETURN);                -- #75



   --------------------------------------------------------------------
   -- ENABLE PROCESSING OF ENDITEMS
   --

   --                        ------------------------
   -- Will enable the processing of ITEM (measurement, software variable,
   -- monitoring list or measurements of a subtree) when the CONDITION_CHECK
   -- of the value of CONDITION_ITEM with CONDITION_VALUE is true (the
   -- condition is true).
   --
   -- If the condition is false, this will disable the processing of ITEM.
   --
   -- By default, all enditems are enabled for processing. Conditions can be
   -- defined to disable and re-enable the processing of given end-items.
   --
   -- When the processing is enabled, ITEM will be authorised for calibration
   -- and monitoring. After ITEM has been enabled, it will be calibrated (if
   -- its acquisition has been started) and monitored (if its monitoring has
   -- been enabled) on reception of new values.
   --
   -- Enabling the processing of ITEM does not perform an "initial"
   -- calibration / monitoring of ITEM in the case where it has already a value.
   --
   -- When the processing is disabled, ITEM will not be calibrated and not be
   -- monitored, even if its acquisition has been requested and if its
   -- monitoring is enabled.
   --
   -- CONDITION_ITEM must be maintained locally otherwise an error is generated.
   -- When the condition is triggered, only those measurements and SW variables
   -- identified by ITEM that are maintained locally will be processed.
   --
   -- It is not possible to set a condition where the ITEM to enable or disable
```

```
-- is the CONDITION_ITEM.
-- If ITEM is a group of enditems (i.e. monitoring list, ADU or incomplete
-- pathname) containing CONDITION_ITEM, the condition will not apply for
-- CONDITION_ITEM.
--
-- It is not possible to enable or disable the processing of a SW variable
-- that is linked to an HK data.
--

-- PARAMETER DESCRIPTION :
-- inputs
-- ------
-- ITEM : item(s) to enable. It can be a measurement or software
--        variable or all measurements / sw variables contained in a
--        monitoring list or all measurements contained in an ADU or
--        all measurements / sw variables contained in a subtree.
--
-- CONDITION_ITEM : the measurement or software variable carrying the
--                  condition.
--
-- CONDITION_CHECK : the check to apply on CONDITION_ITEM.
--
-- CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.
--
-- SINGLE_SHOT     : if true, the condition will be withdrawn after having
--                   being true.
--
-- outputs
-- -------
-- CONDITION_REF   : a unique identifier for the condition
--
-- STATUS : the return status of the operation
--    OK                   : operation successfull
--    INVALID_TESTNODE_MODE : the test node is not in executing mode
--    INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally
--    INVALID_PARAMETER : ITEM and CONDITION_ITEM are identical
--                        or ITEM is a SW variable linked to an HK data.
--                        or CONDITION_CHECK is IN_RANGE (not supported: see above)
--    RUNTIME_ERROR     : an unexpected error occurred

procedure ENABLE_ON_INTEGER
            (in  ITEM            : ACQUISITION_COLLECTION;
             in  CONDITION_ITEM  : INTEGER_MONITOR_ITEM_NAME;
             in  CONDITION_CHECK : CONDITION;
             in  CONDITION_VALUE : INTEGER;
             in  SINGLE_SHOT     : BOOLEAN;
             out CONDITION_REF   : INTEGER;
             out STATUS          : UCL_RETURN);                -- #76
```

```
procedure ENABLE_ON_FLOAT
            (in  ITEM            : ACQUISITION_COLLECTION;
             in  CONDITION_ITEM  : FLOAT_MONITOR_ITEM_NAME;
             in  CONDITION_CHECK : CONDITION;
             in  CONDITION_VALUE : REAL;
             in  SINGLE_SHOT     : BOOLEAN;
             out CONDITION_REF   : INTEGER;
             out STATUS          : UCL_RETURN);               -- #77

procedure ENABLE_ON_STATECODE
            (in  ITEM            : ACQUISITION_COLLECTION;
             in  CONDITION_ITEM  : DISCRETE_MONITOR_ITEM_NAME;
             in  CONDITION_CHECK : DISCRETE_CONDITION;
             in  CONDITION_VALUE : STATECODE;
             in  SINGLE_SHOT     : BOOLEAN;
             out CONDITION_REF   : INTEGER;
             out STATUS          : UCL_RETURN);               -- #78

procedure ENABLE_ON_BYTE_STREAM
            (in  ITEM            : ACQUISITION_COLLECTION;
             in  CONDITION_ITEM  : BYTE_STREAM_MONITOR_ITEM_NAME;
             in  CONDITION_CHECK : DISCRETE_CONDITION;
             in  CONDITION_VALUE : STRING;
             in  SINGLE_SHOT     : BOOLEAN;
             out CONDITION_REF   : INTEGER;
             out STATUS          : UCL_RETURN);               -- #79



--                        ------------------------
-- Will enable or disable the processing of ITEM. This is overwriting
-- an eventual action from a condition.
--

-- PARAMETER DESCRIPTION :
-- inputs
-- ------
-- ITEM : item(s) to enable or disable. It can be a measurement or
--        software variable or all measurements / sw variables contained
--        in a monitoring list or all measurements contained in an ADU or
--        all measurements / sw variables contained in a subtree.
--
-- SWITCH : ON to enable the processing and OFF to disable the processing.
--
-- outputs
-- -------
-- STATUS : the return status of the operation
--    OK                 : operation successfull
--    INVALID_TESTNODE_MODE : the test node is not in executing mode
```

```
--    INVALID_ITEM_NAME : ITEM is not managed locally
--    INVALID_PARAMETER : ITEM is a SW variable linked to an HK data.
--                        or CONDITION_CHECK is IN_RANGE (not supported: see above)
--    RUNTIME_ERROR     : an unexpected error occurred

procedure SET_PROCESSING
            (in  ITEM   : ACQUISITION_COLLECTION;
             in  SWITCH : ON_OFF;
             out STATUS : UCL_RETURN);              -- #80



--                        ------------------------
-- Indicates if a measurement or software variabled is enabled or disabled
-- for processing (as a result of a condition or of a call to SET_PROCESSING)
--

-- PARAMETER DESCRIPTION :
-- inputs
-- ------
-- ITEM : item to check. It can be a single measurement or software
--        variable.
--
-- outputs
-- -------
-- STATE : ON if the processing is enabled and OFF if the processing is
-- disabled.
--
-- STATUS : the return status of the operation
--    OK                : operation successfull
--    INVALID_TESTNODE_MODE : the test node is not in executing mode
--    INVALID_ITEM_NAME : ITEM is not managed locally
--    RUNTIME_ERROR     : an unexpected error occurred

procedure GET_PROCESSING_STATE
            (in  ITEM   : MONITOR_ITEM_NAME;
             out STATE  : ON_OFF;
             out STATUS : UCL_RETURN);              -- #81

--------------------------------------------------------------------



--------------------------------------------------------------------
-- SWITCH LIMIT SET
--

--                        ------------------------
-- Will set the LIMIT_SET for ITEM (measurement or software variable) when
```

```
-- the CONDITION_CHECK of the value of CONDITION_ITEM with CONDITION_VALUE
-- is true (the condition is true).
--
-- This will not perform an "initial" monitoring of ITEM in the case where
-- it has already a value. The monitoring will be performed on reception of
-- the next value (in case the monitoring has been enabled).
-- This can lead to a temporary "inconsistent" output of the procedure
-- GET_FULL_ENDITEM_MONITOR_STATUS (result of the monitoring with the old
-- limit set with values of the new limit set might be inconsistent).
--
-- ITEM and CONDITION_ITEM must be maintained locally on the same node
-- otherwise an error is generated.
--
-- It is not possible to set a condition where the ITEM identical to the
-- CONDITION_ITEM.
--
-- The LIMIT_SET must be defined for ITEM, otherwise an error is generated
--

-- PARAMETER DESCRIPTION :
-- inputs
-- ------
-- ITEM : item for which the limit set is to be switched. It can be a
--        measurement or software variable.
--
-- LIMIT_SET : the limit set number to select.
--
-- CONDITION_ITEM : the measurement or software variable carrying the
--                  condition.
--
-- CONDITION_CHECK : the check to apply on CONDITION_ITEM.
--
-- CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.
--
-- SINGLE_SHOT     : if true, the condition will be withdrawn after having
--                   being true.
--
-- outputs
-- -------
-- CONDITION_REF   : a unique identifier for the condition
--
-- STATUS : the return status of the operation
--    OK                  : operation successfull
--    INVALID_TESTNODE_MODE : the test node is not in executing mode
--    INVALID_ITEM_NAME : ITEM or CONDITION_ITEM are not managed locally
--    INVALID_PARAMETER : ITEM and CONDITION_ITEM are identical
--                        or CONDITION_CHECK is IN_RANGE (not supported: see above)
--    INVALID_LIMIT_SET : the limit set is not defined for ITEM
--    RUNTIME_ERROR     : an unexpected error occurred
```

```
procedure SET_LIMIT_SET_ON_INTEGER
            (in  ITEM            : MONITOR_COLLECTION;
             in  LIMIT_SET       : LIMIT_SET_NUMBER;
             in  CONDITION_ITEM  : INTEGER_MONITOR_ITEM_NAME;
             in  CONDITION_CHECK : CONDITION;
             in  CONDITION_VALUE : INTEGER;
             in  SINGLE_SHOT     : BOOLEAN;
             out CONDITION_REF   : INTEGER;
             out STATUS          : UCL_RETURN);              -- #82

procedure SET_LIMIT_SET_ON_FLOAT
            (in  ITEM            : MONITOR_COLLECTION;
             in  LIMIT_SET       : LIMIT_SET_NUMBER;
             in  CONDITION_ITEM  : FLOAT_MONITOR_ITEM_NAME;
             in  CONDITION_CHECK : CONDITION;
             in  CONDITION_VALUE : REAL;
             in  SINGLE_SHOT     : BOOLEAN;
             out CONDITION_REF   : INTEGER;
             out STATUS          : UCL_RETURN);              -- #83

procedure SET_LIMIT_SET_ON_STATECODE
            (in  ITEM            : MONITOR_COLLECTION;
             in  LIMIT_SET       : LIMIT_SET_NUMBER;
             in  CONDITION_ITEM  : DISCRETE_MONITOR_ITEM_NAME;
             in  CONDITION_CHECK : DISCRETE_CONDITION;
             in  CONDITION_VALUE : STATECODE;
             in  SINGLE_SHOT     : BOOLEAN;
             out CONDITION_REF   : INTEGER;
             out STATUS          : UCL_RETURN);              -- #84

procedure SET_LIMIT_SET_ON_BYTE_STREAM
            (in  ITEM            : MONITOR_COLLECTION;
             in  LIMIT_SET       : LIMIT_SET_NUMBER;
             in  CONDITION_ITEM  : BYTE_STREAM_MONITOR_ITEM_NAME;
             in  CONDITION_CHECK : DISCRETE_CONDITION;
             in  CONDITION_VALUE : STRING;
             in  SINGLE_SHOT     : BOOLEAN;
             out CONDITION_REF   : INTEGER;
             out STATUS          : UCL_RETURN);              -- #85

---------------------------------------------------------------------



---------------------------------------------------------------------
-- START AUTOMATED PROCEDURE
--
```

```
--                        ------------------------
-- Will start an automated procedure when the CONDITION_CHECK of the value
-- of CONDITION_ITEM with CONDITION_VALUE is true (the condition is true).
--
-- CONDITION_ITEM must be maintained locally on the node otherwise an
-- error is generated.
--
-- The AP to start must be an AP without parameters, otherwise an error
-- message is generated when the condition is triggered.
-- When the condition is triggered, if the AP is already running, it will
-- not be started and an error message will be generated.
--
--

-- PARAMETER DESCRIPTION :
-- inputs
-- ------
-- AP : the name of the automated procedure to start. This must be an AP
--      without parameter or where all parameters have default values.
--
-- CONDITION_ITEM : the measurement or software variable carrying the
--                  condition.
--
-- CONDITION_CHECK : the check to apply on CONDITION_ITEM.
--
-- CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.
--
-- SINGLE_SHOT     : if true, the condition will be withdrawn after having
--                   being true.
--
-- outputs
-- -------
-- CONDITION_REF   : a unique identifier for the condition
--
-- STATUS : the return status of the operation
--    OK                 : operation successfull
--    INVALID_TESTNODE_MODE : the test node is not in executing mode
--    INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally
--    INVALID_PARAMETER : CONDITION_CHECK is IN_RANGE (not supported: see above)
--    RUNTIME_ERROR     : an unexpected error occurred

procedure START_AP_ON_INTEGER
            (in  AP              : AP_NAME;
             in  CONDITION_ITEM  : INTEGER_MONITOR_ITEM_NAME;
             in  CONDITION_CHECK : CONDITION;
             in  CONDITION_VALUE : INTEGER;
             in  SINGLE_SHOT     : BOOLEAN;
             out CONDITION_REF   : INTEGER;
             out STATUS          : UCL_RETURN);              -- #86
```

```
procedure START_AP_ON_FLOAT
            (in  AP              : AP_NAME;
             in  CONDITION_ITEM  : FLOAT_MONITOR_ITEM_NAME;
             in  CONDITION_CHECK : CONDITION;
             in  CONDITION_VALUE : REAL;
             in  SINGLE_SHOT     : BOOLEAN;
             out CONDITION_REF   : INTEGER;
             out STATUS          : UCL_RETURN);               -- #87

procedure START_AP_ON_STATECODE
            (in  AP              : AP_NAME;
             in  CONDITION_ITEM  : DISCRETE_MONITOR_ITEM_NAME;
             in  CONDITION_CHECK : DISCRETE_CONDITION;
             in  CONDITION_VALUE : STATECODE;
             in  SINGLE_SHOT     : BOOLEAN;
             out CONDITION_REF   : INTEGER;
             out STATUS          : UCL_RETURN);               -- #88

procedure START_AP_ON_BYTE_STREAM
            (in  AP              : AP_NAME;
             in  CONDITION_ITEM  : BYTE_STREAM_MONITOR_ITEM_NAME;
             in  CONDITION_CHECK : DISCRETE_CONDITION;
             in  CONDITION_VALUE : STRING;
             in  SINGLE_SHOT     : BOOLEAN;
             out CONDITION_REF   : INTEGER;
             out STATUS          : UCL_RETURN);               -- #89

--------------------------------------------------------------------



--------------------------------------------------------------------
-- REMOVE CONDITION
--

--                        ------------------------
-- Will remove a condition or all conditions of a CONDITION_ITEM.

-- PARAMETER DESCRIPTION :
-- inputs
-- ------
-- CONDITION_ITEM : the measurement or software variable carrying the
--                  condition.
--
-- CONDITION_REF   : the unique identifier of the condition to remove
--                   or the constant ALL_CONDITIONS to remove all conditions
--                   of CONDITION_ITEM.
--
-- outputs
```

```
-- -------
-- STATUS : the return status of the operation
--    OK                 : operation successfull
--    INVALID_TESTNODE_MODE : the test node is not in executing mode
--    INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally or
--                        CONDITION_REF does not identify an existing
--                        condition
--    RUNTIME_ERROR      : an unexpected error occurred

procedure WITHDRAW_CONDITION
            (in  CONDITION_ITEM : MONITOR_ITEM_NAME;
             in  CONDITION_REF  : INTEGER := ALL_CONDITIONS;
             out STATUS         : UCL_RETURN);                -- #90



--                        ------------------------
-- Will remove all conditions defined on the test node.

-- PARAMETER DESCRIPTION :
-- outputs
-- -------
-- STATUS : the return status of the operation
--    OK                 : operation successfull
--    INVALID_TESTNODE_MODE : the test node is not in executing mode
--    RUNTIME_ERROR      : an unexpected error occurred

procedure WITHDRAW_ALL_CONDITIONS
            (out STATUS : UCL_RETURN);                         -- #91

--------------------------------------------------------------------





--------------------------------------------------------------------
-- MANAGE CONDITIONS
--


--                        ------------------------
-- Returns the number of items carrying conditions (CONDITION_ITEMS).
-- This number is as well maintained as Housekeeping variable (and can
-- be mapped to a software variable).

function NUMBER_OF_CONDITION_ITEMS : INTEGER;                  -- #92



--                        ------------------------
-- Provide the name of the CONDITION_ITEM identified by an index.
```

```
–– To get all the CONDITION_ITEM’s, this procedure has to be called
–– with all numbers betwee 1 and the value of NUMBER_OF_CONDITION_ITEMS.
––
–– Note:
–– it can occur that between the time where NUMBER_OF_CONDITION_ITEMS has
–– been called and the time where GET_CONDITION_ITEM is called, the number
–– of items carrying condition is modified (if new conditions are created
–– or removed in parallel).

–– PARAMETER DESCRIPTION :
–– inputs
–– ––––––
–– ITEM_NUMBER : index of the CONDITION_ITEM (between 1 and the value of
––               NUMBER_OF_CONDITION_ITEMS).
––
–– outputs
–– –––––––
–– CONDITION_ITEM : the measurement or software variable carrying the
––                  condition.
––
–– STATUS : the return status of the operation
––    OK                 : operation successfull
––    INVALID_TESTNODE_MODE : the test node is not in executing mode
––    INVALID_PARAMETER : There is no (more) CONDITION_ITEM at the index
––                        ITEM_NUMBER (see above note).
––    RUNTIME_ERROR     : an unexpected error occurred

procedure GET_CONDITION_ITEM
            (in  ITEM_NUMBER    : INTEGER;
             out CONDITION_ITEM : MONITOR_ITEM_NAME;
             out STATUS         : UCL_RETURN);                  –– #93


––                          ––––––––––––––––––––––––
–– Provide the number of conditions carried by a CONDITION_ITEM.

–– PARAMETER DESCRIPTION :
–– inputs
–– ––––––
–– CONDITION_ITEM : the name of the item for which the number of conditions
––                  is to be returned. It must be managed locally, otherwise
––                  the value 0 is returned.

function NUMBER_CONDITIONS
            (in  CONDITION_ITEM : MONITOR_ITEM_NAME) :INTEGER;    –– #94


––                          ––––––––––––––––––––––––
–– Provide the description of a condition.
```

```
--
-- Note:
-- it can occur that between the time where GET_NUMBER_CONDITIONS has
-- been called and the time where GET_CONDITION is called, the number
-- of condition carried by CONDITION_ITEM is modified
-- (if new conditions are created or removed in parallel).

-- PARAMETER DESCRIPTION :
-- inputs
-- ------
-- CONDITION_ITEM : the name of the item carrying the condition.
--
-- CONDITION_NUMBER : the index of the condition (between 1 and the value
--                    of GET_NUMBER_CONDITIONS).
--
-- outputs
-- -------
-- CONDITION_REF : the unique identifier of the condition.
--
-- CONDITION_CHECK : the check to apply on CONDITION_ITEM.
--
-- CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.
--
-- SINGLE_SHOT : indicates if the condition is to be removed after having
--               being true.
--
-- STATE : indicate the state of the condition. It can be
--         IS_TRUE : the condition is true, the associated action has been
--                   triggered
--         IS_FALSE : the condition is false, the associated action has been
--                    triggered (only for the enable/disable conditions)
--         IS_UNKNOWN : the condition has been set but has not yet been
--                      analysed (CONDITION_ITEM has not a valid value)
--
-- ACTION : provide the description of the condition, i.e:
--            - the sid of the item(s) for which the processing is to be
--              enabled
--            - the sid of the item for which a new limit set is to be set
--              and the limit set number
--            - the sid of an AP
--
-- STATUS : the return status of the operation
--    OK                 : operation successfull
--    INVALID_TESTNODE_MODE : the test node is not in executing mode
--    INVALID_PARAMETER : There is no (more) condition at the index
--                        CONDITION_NUMBER (see above note).
--    INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally.
--    RUNTIME_ERROR     : an unexpected error occurred
```

```
   procedure GET_CONDITION
               (in  CONDITION_ITEM   : MONITOR_ITEM_NAME;
                in  CONDITION_NUMBER : INTEGER;
                out CONDITION_REF    : INTEGER;
                out CONDITION_CHECK  : CONDITION;
                out CONDITION_VALUE  : CONDITION_VAL;
                out SINGLE_SHOT      : BOOLEAN;
                out STATE            : CONDITION_STATE;
                out ACTION           : ACTION_DESCRIPTION;
                out STATUS           : UCL_RETURN);                -- #95

   --------------------------------------------------------------------


------------------------------------------------------------------------
-- General Conversion
------------------------------------------------------------------------


procedure PATHNAME_TO_STRING (in PATH : PATHNAME;
                          out NAME: STRING);                    -- #96

  -- convert the pathname type into a string type
  -- If PATH is not existing, an empty string is returned
  -- If NAME is too short to hold resulting string, the string is truncated

function PATH(in NAME: STRING) : PATHNAME;                      -- #97

  -- convert the string type into a pathname type
  -- returns Null-Pathname in case the path is not existing


procedure STATECODE_TO_STRING (in CODE : STATECODE;
                          out CODE_TEXT: STRING);               -- #98

  -- convert the statecode type into a string type
  -- CODE_TEXT should have a length of 8
  -- If CODE_TEXT is too short to hold resulting string, the string is truncated
  -- If CODE_TEXT is longer than 8 characters, it is filled with blanks

function CODE(in CODE_TEXT : STRING) : STATECODE;               -- #99

  -- convert the string type into a statecode type
  -- If CODE_TEXT is longer than 8 characters, the string is truncated
  -- If CODE_TEXT is shorter than 8 characters, the STATECODE is extended by blanks


END GROUND_LIBRARY;
```

I-1.3 Interface Description

I-1.3.1 Monitoring

I-1.3.1.1 ENABLE_MONITORING

Specification

procedure ENABLE_MONITORING
(in ITEM: MONITOR_COLLECTION;
in LIMIT_SET: LIMIT_SET_NUMBER : = DEFAULT_LIMIT_SET;
in ADU: ADU_NAME := \\;
out STATUS: UCL_RETURN);

Description / Parameters

Enables monitoring of enditem(s).

ITEM : The item to be enabled, which can be

- a single enditem of type MEASUREMENT or SW variable;
- a incomplete pathname pointing to a virtual node, thus enabling all MEASUREMENTs or SW variables in that subtree;
- a monitor list, thus enabling all items in that list.

LIMIT_SET (optional) : the limit set to be used. In case the value is not given or the value given is DEFAULT_LIMIT_SET (0), the selected limit set will be the currently selected limit set. Upon loading of the database, the selected limit set is the limit set 1. This can be changed by calling the operation SET_LIMIT_SET.

ADU (optional) : the ADU Descriptor to be applied. If ITEM is already acquired, this parameter will be ignored. If there is only one ADU Description defined for ITEM, the parameter may be omitted. If more than one is defined and the parameter is omitted, the command enables monitoring by starting acquisition for all.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

ENABLE_MONITORING (\apm\dms, 1, stat);

I-1.3.1.2 DISABLE_MONITORING

Specification

```
procedure  DISABLE_MONITORING
                 (in ITEM: MONITOR_COLLECTION;
                  out STATUS: UCL_RETURN);
```

Description / Parameters

Disables monitoring of enditems.

ITEM : The item to be disabled, which can be

- a single enditem of type MEASUREMENT or SW variable;
- a incomplete pathname pointing to a virtual node, thus enabling all MEASUREMENTs or SW variables in that subtree;
- a monitor list, thus disabling all items in that list.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

DISABLE_MONITORING (\apm\dms, stat);

I-1.3.1.3 SET_HIGH_LIMIT

Specification

```
procedure  SET_HIGH_LIMIT
                 (in ITEM: ANALOG_MONITOR_ITEM_NAME;
                  in LIMIT: REAL;
                  out STATUS: UCL_RETURN);
```

Description / Parameters

Changes the soft high limit of an enditem in the currently selected limit set. If the item is an integer measurement or integer SW variable, LIMIT is rounded to the nearest integer.

ITEM : name of item to have its limit changed.

LIMIT : value to set high limit to.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

SET_HIGH_LIMIT (\apm\dms\cpu_load, 80.0,   stat);

I-1.3.1.4 SET_INTEGER_HIGH_LIMIT

Specification

procedure SET_INTEGER_HIGH_LIMIT
(in ITEM: INTEGER_MONITOR_ITEM_NAME;
in LIMIT: INTEGER;
out STATUS: UCL_RETURN)

Description / Parameters

Changes the soft high limit of an integer enditem (measurement or software variable) in the currently selected limit set.

ITEM : name of item to have its limit changed.

LIMIT : value to set high limit to.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Example Call

SET_INTEGER_HIGH_LIMIT (\apm\dms\counter, 80, stat);

I-1.3.1.5 SET_LOW_LIMIT

Specification

procedure SET_LOW_LIMIT
(in ITEM: ANALOG_MONITOR_ITEM_NAME;
in LIMIT: REAL;
out STATUS: UCL_RETURN);

Description / Parameters

Changes the soft low limit of an enditem in the currently selected limit set. If the item is an integer measurement or integer SW variable, LIMIT is rounded to the nearest integer.

ITEM : name of item to have its limit changed.

LIMIT : value to set low limit to.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Example Call

SET_LOW_LIMIT (\apm\dms\cpu_load, 10.0, stat);

I-1.3.1.6 SET_INTEGER_LOW_LIMIT

Specification

```
;procedure  SET_INTEGER_LOW_LIMIT
                (in ITEM: INTEGER_MONITOR_ITEM_NAME;
                in LIMIT: INTEGER;
                out STATUS: UCL_RETURN)
```

Description / Parameters

Changes the soft low limit of an integer enditem (measurement or software variable) in the currently selected limit set.

ITEM : name of item to have its limit changed.

LIMIT : value to set high limit to.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Example Call

SET_INTEGER_LOW_LIMIT (\apm\dms\counter, 20, stat);

I-1.3.1.7 SET_DELTA_LIMIT

Specification

```
procedure  SET_DELTA_LIMIT
                (in ITEM: ANALOG_MONITOR_ITEM_NAME;
                in LIMIT: REAL;
                out STATUS: UCL_RETURN);
```

Description / Parameters

Changes the soft delta limit of an enditem in the currently selected limit set. If the item is an integer measurement or integer SW variable, LIMIT is rounded to the nearest integer.

ITEM : name of item to have its limit changed.

LIMIT : value to set delta limit to.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Example Call

SET_DELTA_LIMIT (\apm\dms\cpu_load, 20.0, stat);

I-1.3.1.8 SET_INTEGER_DELTA_LIMIT

Specification

```
procedure  SET_INTEGER_DELTA_LIMIT
                    (in ITEM: INTEGER_MONITOR_ITEM_NAME;
                    in LIMIT: INTEGER;
                    out STATUS: UCL_RETURN)
```

Description / Parameters

Changes the soft delta limit of an integer enditem (measurement or software variable) in the currently selected limit set.

ITEM : name of item to have its limit changed.

LIMIT : value to set high limit to.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

SET_INTEGER_DELTA_LIMIT (\apm\dms\counter, 10, stat);

I-1.3.1.9 SET_EXCEPTION_COUNT

Specification

```
procedure  SET_EXCEPTION_COUNT
                    (in ITEM: MONITOR_ITEM_NAME;
                    in N_COUNT: INTEGER;
                    out STATUS: UCL_RETURN);
```

Description / Parameters

Changes the exception count value of a measurement or SW variable.

ITEM : name of item to have its exception count changed.

N_COUNT : integer value to set exception count to.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

SET_EXCEPTION_COUNT (\apm\dms\status, 5, stat);

I-1.3.1.10 SET_EXPECTED_STATE

Specification

```
procedure  SET_EXPECTED_STATE
                    (in ITEM: DISCRETE_MONITOR_ITEM_NAME;
                    in STATE: STATECODE;
                    out STATUS: UCL_RETURN);
```

Description / Parameters

Changes the expected state of a discrete measurement or SW variable.

ITEM : name of item to have its expected state changed.

STATE : state code for desired expected state of item.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

SET_EXPECTED_STATE (\apm\dms\status, $OFF, stat);

I-1.3.1.11 SET_EXPECTED_VALUE

Specification

procedure SET_EXPECTED_VALUE
(in ITEM: BYTE_STREAM_MONITOR_ITEM_NAME;
in VALUE: STRING;
out STATUS: UCL_RETURN);

Description / Parameters

Changes the expected value of a byte stream measurement or SW variable.

ITEM : name of item to have its expected value changed.

VALUE : string containing desired expected value of item.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

SET_EXPECTED_VALUE (\apm\dms\message, ”Hallo ESA”, stat);

I-1.3.1.12 SET_LIMIT_SET

Specification

procedure SET_LIMIT_SET
(in ITEM: MONITOR_COLLECTION;
in LIMIT_SET: LIMIT_SET_NUMBER;
out STATUS: UCL_RETURN);

Description / Parameters

Changes the current limit_set of a measurement or SW variable. Any old or other limit set is ’switched off’ and the newly selected one is used hereafter.In case a virtual path is specified, all measurements, SW variables or derived values under this path get the new limit set number. For items, where the new limit set is not defined, no change is made. The same applies to monitoring lists.

ITEM : name of item to have its limit set changed.

LIMIT_SET : identifier of the desired limit set. Its value must be between 1 and the number of limits defined for ITEM.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

SET_LIMIT_SET (\apm\dms\memory_pages, 4, stat);

I-1.3.1.13 GET_ENDITEM_MONITOR_STATUS

Specification

```
procedure  GET_ENDITEM_MONITOR_STATUS
                (in ITEM: MONITOR_ITEM_NAME;
                 out MONITORING_STATUS: MONITOR_STATUS;
                 out STATUS: UCL_RETURN);
```

Description / Parameters

Returns the current monitoring status for an enditem.

ITEM : name of enditem to monitor.

MONITORING_STATUS : returns monitoring status of Item.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

GET_ENDITEM_MONITOR_STATUS (\apm\power\consumption, enditem_status, stat);

I-1.3.1.14 GET_FULL_ENDITEM_MONITOR_STATUS

Specification

```
procedure  GET_FULL_ENDITEM_MONITOR_STATUS
                (in  ITEM: MONITOR_ITEM_NAME;
                 out MONITORING_STATUS: FULL_MONITOR_STATUS;
                 out STATUS: UCL_RETURN);
```

Description / Parameters

Returns the current monitoring status and monitoring values for an enditem.

ITEM : pathname of enditem to monitor.

MONITOR_STATUS : returns the detailed monitoring status record of Item, containing alternative variants depending on the item type and whether a limit set is defined (see the definition of type `FULL_MONITOR_STATUS`.)

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

GET_FULL_ENDITEM MONITOR_STATUS
(\apm\power\consumption, monitor_status, stat);

I-1.3.1.15 GET_ACQUISITION_STATUS

```
procedure GET_ACQUISITION_STATUS
            (in  ITEM: MONITOR_ITEM_NAME;
             out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
             out STATUS: UCL_RETURN);
```

Description / Parameters

Returns the current acquisition status for an enditem. This operation can be called locally on the test node maintaining the enditem or remotely on another test node.

ITEM : pathname of enditem (measurement or software variable).

ENDITEM_ACQ_STATUS : returns the acquisition status of item.
VALID: The enditem has been startet for acquisition and a value has been delivered
NOT_ACQ : The measurement has not been started for acquisition
NOT_RECVD : The measurement has been started for acquisition but no value has been received up to now
NOT_MAINTAINED: The enditem is not maintained by any test node
DISAB : The measurement / sw variable has been ”disabled” by a condition
INVALID : No valid value exist for the enditem, e.g. it could not be calibrated or it is not maintained by any test node
NOT_AVAILABLE : The SAS has indicated that it cannot deliver any more the end–item
STATIC : The SAS has indicated that it currently cannot deliver the end–item

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
GET_ACQUISITION_STATUS (\apm\power\consumption, acq_status, stat);
IF acqu_status = VALID
                then my_var := \apm\power\consumption;
END IF;
```

I-1.3.1.16 GET_INTEGER

```
procedure GET_INTEGER
            (in  ITEM: INTEGER_MONITOR_ITEM_NAME;
             out VALUE: INTEGER;
             out TIME_TAG: TIME;
             out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
             out STATUS: UCL_RETURN);
```

Description / Parameters

Returns the current value for an integer enditem. This operation can be called locally on the test node maintaining the enditem or remotely on another test node.

ITEM : pathname of an integer enditem (measurement or software variable).

VALUE : the value of the enditem. In case the value is not VALID (see ENDITEM_ACQ_STATUS), the last valid value will be returned (or a default null value in case the value has never been set)

TIME_TAG : the time tag (local time) of the last value. In case the value is not VALID (see ENDITEM_ACQ_STATUS) the time tag of the last valid value will be returned (or a default null time in case the value has never been set)

ENDITEM_ACQ_STATUS : returns the acquisition status of item.
VALID: The enditem has been startet for acquisition and a value has been delivered
NOT_ACQ : The measurement has not been started for acquisition
NOT_RECVD : The measurement has been started for acquisition but no value has been received up to now
NOT_MAINTAINED: The enditem is not maintained by any test node
DISAB : The measurement / sw variable has been "disabled" by a condition
INVALID : No valid value exist for the enditem, e.g. it could not be calibrated or it is not maintained by any test node
STATIC : The SAS has indicated that it cannot deliver any more the end–item

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
GET_INTEGER (\apm\power\counter, int_value, time_tag, acq_status, stat);
```

I-1.3.1.17 GET_FLOAT

```
procedure GET_FLOAT
            (in  ITEM: FLOAT_MONITOR_ITEM_NAME;
             out VALUE: REAL;
             out TIME_TAG: TIME;
             out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
             out STATUS: UCL_RETURN);
```

Description / Parameters

Returns the current value for a float enditem. This operation can be called locally on the test node maintaining the enditem or remotely on another test node.

ITEM : pathname of a float enditem (measurement or software variable).

VALUE : the value of the enditem. In case the value is not VALID (see ENDITEM_ACQ_STATUS), the last valid value will be returned (or a default null value in case the value has never been set)

TIME_TAG : the time tag (local time) of the last value. In case the value is not VALID (see ENDITEM_ACQ_STATUS) the time tag of the last valid value will be returned (or a default null time in case the value has never been set)

ENDITEM_ACQ_STATUS : returns the acquisition status of item.
VALID: The enditem has been startet for acquisition and a value has been delivered
NOT_ACQ : The measurement has not been started for acquisition
NOT_RECVD : The measurement has been started for acquisition but no value has been received up to now
NOT_MAINTAINED: The enditem is not maintained by any test node
DISAB : The measurement / sw variable has been ”disabled” by a condition
INVALID : No valid value exist for the enditem, e.g. it could not be calibrated or it is not maintained by any test node
STATIC : The SAS has indicated that it cannot deliver any more the end–item

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
GET_float (\apm\power\consumption, flt_value, time_tag, acq_status, stat);
```

I-1.3.1.18 GET_STATECODE

```
procedure GET_STATE_CODE
            (in  ITEM: DISCRETE_MONITOR_ITEM_NAME;
             out VALUE: STATECODE;
             out TIME_TAG: TIME;
             out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
             out STATUS: UCL_RETURN);
```

Description / Parameters

Returns the current value for a discrete enditem. This operation can be called locally on the test node maintaining the enditem or remotely on another test node.

ITEM : pathname of a discrete enditem (measurement or software variable).

VALUE : the value of the enditem. In case the value is not VALID (see ENDITEM_ACQ_STATUS), the last valid value will be returned (or a default null value in case the value has never been set)

TIME_TAG : the time tag (local time) of the last value. In case the value is not VALID (see ENDITEM_ACQ_STATUS) the time tag of the last valid value will be returned (or a default null time in case the value has never been set)

ENDITEM_ACQ_STATUS : returns the acquisition status of item.
VALID: The enditem has been startet for acquisition and a value has been delivered
NOT_ACQ : The measurement has not been started for acquisition
NOT_RECVD : The measurement has been started for acquisition but no value has been received up to now
NOT_MAINTAINED: The enditem is not maintained by any test node
DISAB : The measurement / sw variable has been ”disabled” by a condition
INVALID : No valid value exist for the enditem, e.g. it could not be calibrated or it is not maintained by any test node
STATIC : The SAS has indicated that it cannot deliver any more the end–item

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
GET_STATECODE (\apm\power\status, disc_value, time_tag, acq_status, stat);
```

I-1.3.1.19 GET_BYTE_STREAM

```
procedure GET_BYTE_STREAM
            (in  ITEM: BYTE_STREAM_MONITOR_ITEM_NAME;
             out VALUE: STRING;
             out TIME_TAG: TIME;
             out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
             out STATUS: UCL_RETURN);
```

Description / Parameters

Returns the current value for a byte stream enditem. This operation can be called locally on the test node maintaining the enditem or remotely on another test node.

ITEM : pathname of a byte–stream enditem (measurement or software variable).

VALUE : the value of the enditem. In case the value is not VALID (see ENDITEM_ACQ_STATUS), the last valid value will be returned (or a default null value in case the value has never been set). If the string given is not large enough, the value is truncated.

TIME_TAG : the time tag (local time) of the last value. In case the value is not VALID (see ENDITEM_ACQ_STATUS) the time tag of the last valid value will be returned (or a default null time in case the value has never been set)

ENDITEM_ACQ_STATUS : returns the acquisition status of item.
VALID: The enditem has been startet for acquisition and a value has been delivered
NOT_ACQ : The measurement has not been started for acquisition
NOT_RECVD : The measurement has been started for acquisition but no value has been received up to now
NOT_MAINTAINED: The enditem is not maintained by any test node
DISAB : The measurement / sw variable has been ”disabled” by a condition
INVALID : No valid value exist for the enditem, e.g. it could not be calibrated or it is not maintained by any test node
NOT_AVAILABLE : The SAS has indicated that it cannot deliver any more the end–item
STATIC : The SAS has indicated that it cannot deliver any more the end–item

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
GET_BYTE_STREAM (\apm\power\message, str_value, time_tag, acq_status, stat);
```

I-1.3.2 Time Management

I-1.3.2.1 CLOCK

Specification

function CLOCK
(in BASE: TIME_BASE: = LOCAL_TIME) : TIME;

Description / Parameters

Returns the actual local time or SMT

BASE (optional) : time_base to use; by default, local time.

Example Call

my_time_variable : = CLOCK;

I-1.3.2.2 DELAY

Specification

procedure DELAY
(in DELAY: DURATION;
in BASE: TIME_BASE:= LOCAL_TIME);

Description / Parameters

Waits the given time interval. Returns nothing.

DELAY : duration to delay, expressed as a fixed–point number in seconds.

BASE : time base to use; by default, local time.

Example Call

DELAY (4.6 [s]);

I-1.3.2.3 WAIT_UNTIL

Specification

procedure WAIT_UNTIL
(in DUE_TIME : TIME;
in BASE: TIME_BASE: = LOCAL_TIME;
out STATUS: UCL_RETURN);

Description / Parameters

Waits until the given time is reached.

DUE_TIME : time to wait until.

BASE (optional) : time base to use; by default, local time.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

WAIT_UNTIL (3:00:00.000, LOCAL_TIME, stat);

I-1.3.2.4 START_SMT

Specification

```
procedure  START_SMT
                    (in VALUE: TIME: ~:~ ;
                    out STATUS: UCL_RETURN);
```

Description / Parameters

Starts Simulation Mean Time (SMT), and optionally sets it to a given value. If SMT is already running, the new VALUE is immediately set, and time is incremented accordingly thereafter.

VALUE (optional) : new SMT; by default current SMT is resumed.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

START_SMT (10.01.95 20:00:00, STATUS : stat);

11.8.3.1 START_SMT_WITH_OFFSET

Specification

```
procedure  START_SMT_WITH_OFFSET
                    (in OFFSET: INTEGER ;
                    out STATUS: UCL_RETURN);
```

Description / Parameters

Starts Simulation Mean Time (SMT) relative to local time. If SMT is already running, the new time acc. to OFFSET is immediately set, and time is incremented accordingly thereafter.

OFFSET : number of seconds relative to Local Time (LOT). A negative number means number of seconds before (earlier) than LOT.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

START_SMT_WITH_OFFSET (3600, STATUS : stat); — start SMT with one hour later than LOT

I-1.3.2.5 STOP_SMT

Specification

```
procedure  STOP_SMT
                    (out  STATUS: UCL_RETURN);
```

Description / Parameters

Stop the running Simulation Mean Time.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

STOP_SMT (stat);

I-1.3.3 GTAP Handling

In the following, the term GTAP (Ground Test Automated Procedure) is used for Automated Procedures running in the Ground System (to differentiate from Automated Procedures running in the Onboard System)

I-1.3.3.1 EXECUTE_AP

Specification

```
procedure  EXECUTE_AP
                (in AP: AP_NAME ();
                in PRIO: PRIORITY: = LOW;
                in NODE: NODE_NAME : = \\;
                out ID: AP_ID;
                out STATUS: UCL_RETURN);
```

Description / Parameters

Starts execution of a GTAP. If the procedure is called from within a GTAP, the called GTAP runs asynchronously unless SYNCHRONISE_WITH_AP is also called.

AP : name of GTAP to run; optionally including its list of parameters.

PRIO (optional) : the GTAP's priority (HIGH or LOW).

NODE (optional) : node on which the GTAP is to be executed: by default the current node.

ID : returns the new GTAP's identifier, if it could be started; otherwise –1.
STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
EXECUTE_AP
                (\apm\dms\start_up(A,B,C), low_priority, \apm\dms, my_AP_ID, exec_stat);
```

I-1.3.3.2 SYNCHRONISE_WITH_AP

Specification

```
procedure SYNCHRONISE_WITH_AP
                (in AP: AP_ID;
                in NODE: NODE_NAME : =  \\;
                out COMPLETION: AP_COMPLETION_STATE;
                out STATUS: UCL_RETURN);
```

Description / Parameters

Forces the calling GTAP to synchronise with the called GTAP (by default, EXECUTE_AP starts the called GTAP asynchronously). SYNCHRONISE_WITH_AP blocks until the called GTAP has finished.

AP : internal GTAP identifier, from EXECUTE_AP.

NODE (optional) : node on which the GTAP is to be executed; by default the current node.

COMPLETION : returns the called GTAP's completion state.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
SYNCHRONISE_WITH_AP (my_AP_ID, \DMS\SCOE,  my_completion_state,  stat);
```

I-1.3.3.3 SUSPEND_AP

Specification

```
procedure  SUSPEND_AP
                  (in AP: AP_ID;
                  in NODE: NODE_NAME : = \\;
                  out STATUS: UCL_RETURN);
```

Description / Parameters

Suspends execution of a GTAP on a given test node.

AP : internal GTAP identifier, given to it when started by EXECUTE_AP.

NODE (optional) : the node running the GTAP : by default, the current node.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

SUSPEND_AP (power_down_AP_ID,\\, stat);

I-1.3.3.4 RESUME_AP

Specification

```
procedure  RESUME_AP
                  (in AP: AP_ID;
                  in NODE: NODE_NAME : = \\;
                  out STATUS: UCL_RETURN);
```

Description / Parameters

Resumes execution of a suspended GTAP.

AP : internal GTAP identifier, given to it when started by EXECUTE_AP.

NODE (optional) : the node running the GTAP : by default the local node.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

RESUME_AP (my_AP_reference, \\, stat);

I-1.3.3.5 OWN_AP_ID

Specification

function OWN_AP_ID : AP_ID

Description / Parameters

Returns the GTAP identifier of the AP calling that function. This function is not available from HLCL as the system library commands executed from HLCL do not have a GTAP identifier.

Example Call

MY_AP_ID := OWN_AP_ID;

I-1.3.3.6 GET_AP_ID

Specification

```
procedure   GET_AP_ID
                (in AP: AP_NAME;
                in NODE: NODE_NAME : = \\;
                out NUMBER: INTEGER;
                out ID_LIST: AP_ID_LIST;
                out STATUS: UCL_RETURN);
```

Description / Parameters

Returns all GTAP identifiers for a given automated procedure name on a given test node. Since several instances of a GTAP may run in parallel on one test node, a list of GTAP identifiers is returned.

AP : name of GTAP to find.

NODE (optional) : node to search; by default, the local node.

NUMBER : returns the number of GTAP identifiers in the ID_List for this GTAP.

ID_LIST : returns list of AP_IDs found. Unused fields of the ID_List are filled with zeroes.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
GET_AP_ID
                (\apm\power\control_program, \apm\dms, my_number_of_APs, my_AP_list, stat);
```

I-1.3.3.7 GET_AP_STATUS

Specification

```
procedure  GET_AP_STATUS
                (in AP: AP_ID;
                in NODE: NODE_NAME : = \\;
                out EXEC_STATE: AP_EXECUTION_STATE;
                out STATUS: UCL_RETURN);
```

Description / Parameters

Returns the current status of a GTAP.

AP : AP_ID of the GTAP whose status is required.

NODE (optional) : node to search; by default, the current node.

EXEC_STATE : returns the execution state of the GTAP.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
GET_AP_STATUS (\apm\power\off\procedure, \apm\dms, my_AP_status,  stat);
```

I-1.3.3.8 READ_MESSAGE_FROM_AP

Specification

```
procedure READ_MESSAGE_FROM_AP
        (in BLOCK: BOOLEAN : = TRUE;
        out AP: AP_ID;
        out NODE_NAME: USER_MESSAGE;
        out MESSAGE : USER_MESSAGE;
        out STATUS: UCL_RETURN);
```

Description / Parameters

Reads a message asynchronously from another GTAP.

BLOCK (optional) : boolean flag controlling the source of the message, as follows:

- TRUE : waits until a GTAP really sends a message (by default)
- FALSE : reads the next message buffered, or if none, returns an empty string and a null AP_ID

AP : returns identifier of the GTAP sending the message.

NODE_NAME : returns string identifying node on which other GTAP is running.

MESSAGE : returns string from the other GTAP.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Example Call

READ_MESSAGE_FROM_AP (FALSE, the_AP_ID, the_AP_message, stat);

I-1.3.3.9 WRITE_MESSAGE_TO_AP

Specification

```
procedure  WRITE_MESSAGE_TO_AP
        (in AP: AP_ID;
        in NODE: NODE_NAME : = \\;
        in MESSAGE: USER_MESSAGE : = DEFAULT_MESSAGE;
        out  STATUS: UCL_RETURN);
```

Description / Parameters

Sends a message asynchronously from one GTAP to another, i.e. without waiting for the receiving AP to accept the message. The caller simply sends the message and continues.

AP : AP_ID of GTAP to receive the message.

NODE (optional) : processor node running receiving GTAP; by default, the current node.

MESSAGE : string to send.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Example Call

WRITE_MESSAGE_TO_AP
        (dms_control_AP_ID, \DMS\SCOE, ”EMERGENCY STOP”, stat);

I-1.3.4 Application Handling

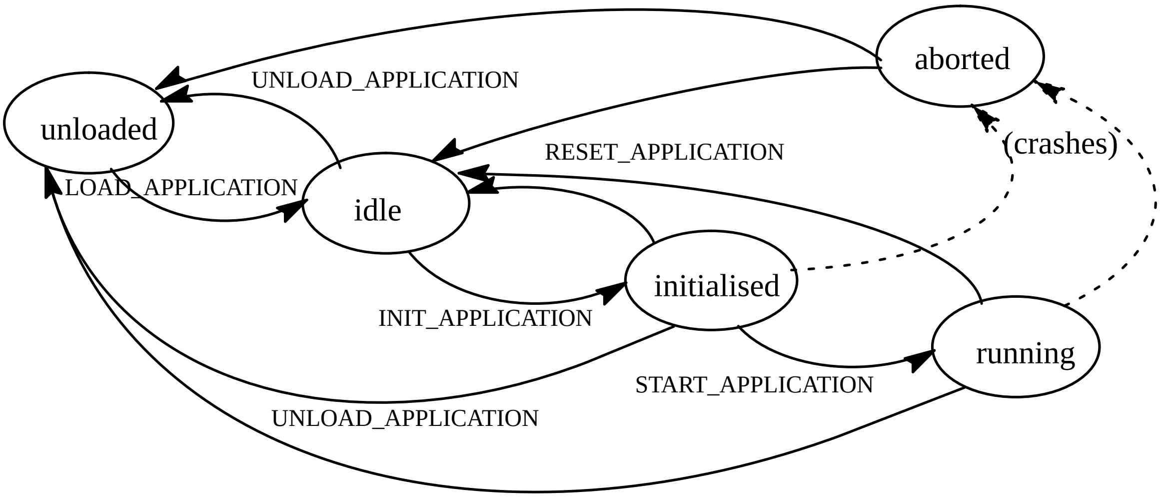


Figure: **Relationship of Application Handling Primitives**

I-1.3.4.1 LOAD_APPLICATION

Specification

procedure LOAD_APPLICATION
(in NAME : APPLICATION_NAME;
in NODE: NODE_NAME := \\;
in PARAMS: USER_MESSAGE : = DEFAULT_MESSAGE;
out APPL: APPLICATION_ID;
out STATUS: UCL_RETURN);

Description / Parameters

Loads an application on to the local test node or on a workstation.

NAME : the name of the executable image in the predefined application directory.

NODE : the name of the node where the SAS shall execute. By default the empty pathname denotes the local testnode. The workstations are referenced by their EGSE node entry in MDA. Starting a SAS on another testnode is prohibited.

PARAMS (optional) : parameter string up to 255 characters to pass to the application.

APPL : returns the application's ID to be used for subsequent control operations.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

LOAD_APPLICATION (”exp_monitor”, ”Exp=123”, my_appl_ID, stat);

I-1.3.4.2 UNLOAD_APPLICATION

Specification

procedure UNLOAD_APPLICATION
(in APPL: APPLICATION_ID;
out STATUS: UCL_RETURN);

Description / Parameters

Unloads the specified application from memory.

APPL : APPLICATION_ID as returned from LOAD_APPLICATION (section I-1.3.4.1).

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

UNLOAD_APPLICATION (smives_appl_ID, stat);

I-1.3.4.3 INIT_APPLICATION

Specification

procedure INIT_APPLICATION
(in APPL: APPLICATION_ID;
in MESSAGE: USER_MESSAGE: = DEFAULT_MESSAGE;
out STATUS: UCL_RETURN);

Description / Parameters

Initialises the specified application.

APPL : APPLICATION_ID as returned from LOAD_APPLICATION (section I-1.3.4.1).

MESSAGE (optional) : string up to 255 characters to send to the application.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

INIT_APPLICATION (my_appl_ID, ”GNC_RECEIVER_ON”, stat);

I-1.3.4.4 START_APPLICATION

Specification

procedure START_APPLICATION
(in APPL: APPLICATION_ID;
out STATUS: UCL_RETURN);

Description / Parameters

Starts the application running.

APPL : APPLICATION_ID as returned from LOAD_APPLICATION (section I-1.3.4.1).

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

START_APPLICATION (my_appl_ID, stat);

I-1.3.4.5 RESET_APPLICATION

Specification

```
procedure  RESET_APPLICATION
                (in APPL: APPLICATION_ID;
                out STATUS: UCL_RETURN);
```

Description / Parameters

Resets the specified application.

APPL : APPLICATION_ID as returned from LOAD_APPLICATION (section I-1.3.4.1).

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

RESET_APPLICATION (my_appl_ID, stat);

I-1.3.4.6 GET_APPLICATION_STATUS

Specification

```
procedure  GET_APPLICATION_STATUS
                (in APPL: APPLICATION_ID;
                out  STATE: APPLICATION_STATE;
                out  ERROR_COUNT: INTEGER;
                out ERROR_MESSAGE: APPLICATION_ERROR_MESSAGE;
                out  STATUS: UCL_RETURN);
```

Description / Parameters

Returns the status of an application, including error status and statistics.

APPL : APPLICATION_ID as returned from LOAD_APPLICATION (section I-1.3.4.1).

STATE : returns state of application.

ERROR_COUNT : returns integer count of errors in the application.

ERROR_MESSAGE : returns error message from the application.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

GET_APPLICATION_STATUS (appl_ID, appl_state, err_count, appl_error_mess, stat);

I-1.3.4.7 GET_APPLICATION_ID

Specification

```
procedure GET_APPLICATION_ID
        (in  NAME: APPLICATION_NAME;
         out APPL: APPLICATION_ID;
         out STATUS: UCL_RETURN);
```

Description / Parameters

Returns the Application ID of an application, if the application is loaded

NAME : The name of the application.

APPL : APPLICATION_ID as returned from LOAD_APPLICATION (section I-1.3.4.1).

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

GET_APPLICATION_ID (appl_name, appl_id, stat);

I-1.3.4.8 GET_APPLICATION_NAME

Specification

```
procedure GET_APPLICATION_NAME
        (in  APPL: APPLICATION_ID;
         out NAME: APPLICATION_NAME;
         out STATUS: UCL_RETURN);
```

Description / Parameters

Returns the name of an application

APPL : APPLICATION_ID as returned from LOAD_APPLICATION (section I-1.3.4.1).

NAME : returns the name of the application.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

GET_APPLICATION_NAME (appl_ID, appl_name, stat);

I-1.3.4.9 READ_MESSAGE_FROM_APPLICATION

Specification

procedure READ_MESSAGE_FROM_APPLICATION
(in BLOCK: BOOLEAN: = TRUE;
out APPL: APPLICATION_ID;
out MESSAGE : USER_MESSAGE;
out STATUS: UCL_RETURN);

Description / Parameters

Reads a message from an application.

BLOCK : boolean flag, controlling the source of the read:

– TRUE, waits until an application really sends a message;

– FALSE, reads the next message buffered. If none, returns an empty string and a null application id.

APPL : returns APPLICATION_ID of the application sending the message.

MESSAGE : returns a string containing the message

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

READ_MESSAGE_FROM_APPLICATION (FALSE, appl_ID, appl_message, stat)

I-1.3.4.10 WRITE_MESSAGE_TO_APPLICATION

Specification

procedure WRITE_MESSAGE_TO_APPLICATION
(in APPL: APPLICATION_ID;
in MESSAGE : USER_MESSAGE;
out STATUS: UCL_RETURN);

Description / Parameters

Sends a message to an application.

APPL : the APPLICATION_ID as returned from LOAD_APPLICATION.

MESSAGE : string of up to 255 characters to be sent.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

WRITE_MESSAGE_TO_APPLICATION (smives_appl_ID, ”IEEE_BUS_1_INIT”, stat);

I-1.3.5 Issue of HW Stimuli

I-1.3.5.1 ISSUE

Specification

```
procedure ISSUE
        (in ITEM: ISSUE_ITEM_NAME();
        in PRIO: PRIORITY := LOW;
        in AT_TIME: TIME: = ~:~;
        in BASE: TIME_BASE: = LOCAL_TIME;
        in ONBOARD_EXECUTION_TIME: TIME := ~:~;
        in TIMEOUT: DURATION: = DEFAULT_ISSUE_TIMEOUT ;
        out STATUS: UCL_RETURN)
```

Description / Parameters

Issues a stimulus (or Telecommand).

ITEM : the database enditem describing what has to be issued. It can denote a parameter list, as defined for the enditem.
Note: the enditem may also be maintained by a remote node

PRIO : the priority to be used for issuing the item (HIGH or LOW)

AT_TIME (optional) : The time to issue the stimulus;

Default: ”~:~” = ”00.00.01”, i.e. the default ”no time”.

The purpose of the AT_TIME parameter shall be the same for all GDU alternatives:

For MDB enditems of type EGSE_ANALOGUE_STIMULUS, EGSE_DISCRETE_STIMULUS and EGSE_PREDEFINED_TC, the time value is written as the TIME_TAG into the GDU generated by the command and sent to the SAS. The SAS then has to extract the TIME_TAG from the GDU and execute the stimulus at the given time.

For MDB enditems of type GDU_DESCRIPTION_LIST the AT_TIME value will be written into the time tasg field of the GDU header of each stimulus referenced within the GDU list.

ONBOARD_EXECUTION_TIME (optional): The time to excute the telecommand onboard

Default: ”~:~” = ”00.00.01”, i.e. the default ”no time”.

For MDB Enditems of type EGSE_PREDEFINED_TC, the ONBOARD_EXECUTION _TIME value will be written to the TIME field of the secondary header of the corresponding CCSDS packet. In this case, the TIME_ID field of the secondary header shall be set to ’10’B, which means: a time tag is present. No interpretation of TIME_BASE will be done by CGS in this case. The TIME_TAG of the GDU header generated by the command will be independent from this.

If no ONBOARD_EXECUTION _TIME parameter is given, i.e. the default value is applied , the TIME_ID and TIME field shall be set to ’10’B (= time tagged command: the execution of the command is scheduled by the onboard system to the specified time)

For MDB enditems of type GDU_DESCRIPTION_LIST the ONBOARD_EXECUTION _TIME value will be written into the TIME field of the secondary header of each CCSDS packet referenced within the GDU list.

Remark: If the TIME_ID is of value ’00’B (UNDEFINED) or ’11’B (NO_TIME_TAG), the TIME field in the secondary header shall be set to zero via the database default entry, in case no ONBOARD_EXECUTION_TIME parameter is given.

BASE (optional) : specifies if AT_TIME is local time or SMT; by default, local time.

TIMEOUT : the maximum time to wait for acknowledgement of this stimulus or telecommand

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

ISSUE (\apm\ecls\main_temperature(21.1), AT_TIME : 10:00:00, STATUS : stat);

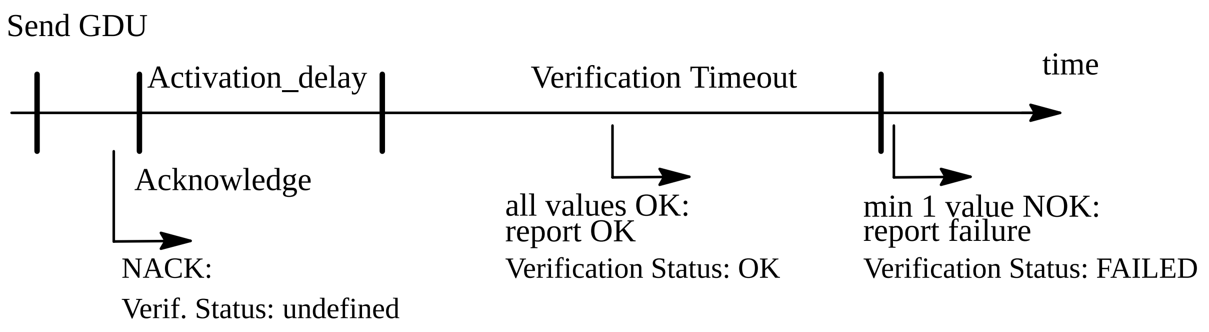
I-1.3.5.2 ISSUE_AND_VERIFY

Specification

```
procedure ISSUE_AND_VERIFY
        (in ITEM: SINGLE_GDU_NAME();
         in PRIO: PRIORITY := LOW;
         in AT_TIME: TIME := ~:~;
         in BASE: TIME_BASE := LOCAL_TIME;
         in ONBOARD_EXECUTION_TIME: TIME := ~:~;
         in TIMEOUT: DURATION := default_issue_timeout;
         in ACTIVATION_DELAY: DURATION := –1.0 [s];
         in VERIFICATION_TIMEOUT : DURATION := –1.0 [s];
         in WAIT_TIL_END: BOOLEAN := FALSE;
         out VERIFICATION_STATUS: CMD_VERIFICATION_STATUS;
         out STATUS: UCL_RETURN);
```

Description / Parameters

Issues a stimulus (or Telecommand/Binary packet) and verifies expected values for measurements, defined as part of the enditem definition in the MDB. The verification starts after the command has been acknowledged and a specified time (ACTIVATION_DELAY) has elapsed. If in the VERIFICATION_TIME-OUT period all specified expected values have been verified, the verification status is set to OK, otherwise to FAILED. A message is generated accordingly.
If during the verification an error occurs, the verification status is set to ERROR.
If WAIT_TIL_END is set, the procedure waits until the end of the verification and the verification status is returned in VERIFICATION_STATUS; otherwise the status can be read by the procedure GET_VERIFICATION_STATUS at any time.



ITEM : the database enditem (GDU) describing what has to be issued. It can denote a parameter list, as defined for the enditem.
Notes: (1) the enditem may also be maintained by a remote node

PRIO : the priority to be used for issuing the GDU (HIGH or LOW)

AT_TIME (optional) : The time to issue the GDU;
—> see procedure ISSUE

ONBOARD_EXECUTION_TIME (optional): The time to execute the telecommand onboard
—> see procedure ISSUE

BASE (optional) : specifies if AT_TIME is local time or SMT; by default, local time.

TIMEOUT : the maximum time to wait for acknowledgement of this GDU

ACTIVATION_DELAY: the time to wait after sending of the GDU until the verification of the measurements is started. If this value is set to 0.0, the checking is performed immediately after the command has been acknowledged. If the default value (–1.0) is given, the activation delay is taken from the MDB definition.

VERIFICATION_TIMEOUT: the time to wait after the ACTIVATION_DELAY has expired and before the verification of the measurements is finished. If in this period the verification of the value(s) for measurement(s) defined in the MDB is successful, a message is generated and the verification status is set to OK. If VERIFICATION_TIMEOUT has elapsed and the verification of at least one measurment is not successful, a message is generated indicating failure and the verification status is set to FAILED.

If VERIFICATION_TIMEOUT is set to 0.0, the checking is performed immediately after the ACTIVATION_DELAY and then is finished.If the default value (–1.0) is given, the activation delay is taken from the MDB definition.

WAIT_TIL_END: If true, the procedure returns after the verification has finished. If false, the procedure returns immediately after the GDU has been acknowledged resp. after TIMEOUT. In this case, the VERIFICATION_STATUS is set to IN_PROGRESS.

VERIFICATION_STATUS: Status of the verification

STARTED,	Verification is started, i.e. Activation Delay is currently running
IN_PROGRESS,	Verification is in progress, i.e Verification Timeout is running and not all measurements have expected values yet
TC_VERIF_OK,	Verification performed with success. All measurements had expected values.
TC_VERIF_FAILED,	Verification performed with no success, i.e. Verification Timeout elapsed and at least one measurement had no expected value
ERROR,	Error during Verification Timeout. During the verification timeout an internal error occured
NO_VERIFICATION	No Verification defined for this enditem (i.e. in MDB there are no measurements specified which to verify for this command)
ABORTED	Verification aborted due to issue of same command or stop of remote TES
UNDEFINED	The verifcation status is not defined, i.e. the sending of the command aborted due to error or negative acknowledge or timeout

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

ISSUE_AND_VERIFY(\apm\eps\power_on, VERIFICATION_TIMEOUT: 3.0[s],WAIT_TIL_END: true,
VERIFICATION_STATUS: v_stat, STATUS : stat);

I-1.3.5.3 GET_VERIFICATION_STATUS

Specification

procedure GET_VERIFICATION_STATUS
(in ITEM: GDU_ITEM_NAME;
out VERIFICATION_STATUS: CMD_VERIFICATION_STATUS;
out STATUS: UCL_RETURN);

ENDITEM : name of stimulus or telecommand for which the verification status is to be fetched. Enditem may be defined on local or remote node.

VERIFICATION_STATUS: Status of the verification. Applies to the last command (GDU) successfully sent or just being sent.

STARTED,	Verification is started, i.e. Activation Delay is currently running
IN_PROGRESS,	Verification is in progress, i.e Verification Timeout is running and not all measurements have expected values yet
TC_VERIF_OK,	Verification performed with success. All measurments had expected values.
TC_VERIF_FAILED,	Verification performed with no success, i.e. Verification Timeout elapsed and at least one measurement had no expected value
ERROR,	Error during Verification Timeout. During the verification timeout an internal error occured
NO_VERIFICATION	No Verification defined for this enditem (i.e. in MDB there are no measurements specified which to verify for this command)
ABORTED	Verification aborted due to issue of same command or stop of remote TES
UNDEFINED	The verifcation status is not defined, i.e. the sending of the command aborted due to error or negative acknowledge or timeout

I-1.3.5.4 ENABLE_ENDITEM

Specification

procedure ENABLE_ENDITEM
(in ENDITEM: ISSUE_ITEM_NAME;
out STATUS: UCL_RETURN);

Description / Parameters

Enables an enditem that has been disabled, either temporarily through DISABLE_ENDITEM, or permanently through definition in the configuration database.

ENDITEM : name of stimulus or telecommand to be enabled.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Example Call

ENABLE_ENDITEM (\apm\power\off, stat);

I-1.3.5.5 DISABLE_ENDITEM

Specification

procedure DISABLE_ENDITEM
(in ENDITEM: ISSUE_ITEM_NAME;
out STATUS: UCL_RETURN);

Description / Parameters

Disables an enditem that has been enabled temporarily or permanently.

ENDITEM : name of stimulus or telecommand to be disabled.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

DISABLE_ENDITEM (\apm\power\off, stat);

I-1.3.5.6 DOWNLOAD

Specification

procedure DOWNLOAD
(in APPL: APPLICATION_ID;
in ITEM: DOWNLOAD_ITEM_NAME;
in NODE: NODE_NAME :=\\;
out STATUS: UCL_RETURN);

Description / Parameters

Downloads a file to a SAS, the corresponding frontend equipment managed by the SAS or the unit under test through a SAS.

APPL : name of the SAS which is doing the download

ITEM : name of the item to be downloaded

NODE : name of the node into which the item has to be downloaded. An empty pathname indicates some kind of default for the SAS.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

DISABLE_ENDITEM (\apm\power\off, stat);

I-1.3.6 Raw Data Handling

I-1.3.6.1 START_ACQUISITION

Specification

```
procedure  START_ACQUISITION
                    (in ITEM: ACQUISITION_COLLECTION;
                    in ADU: ADU_NAME := \\;
                    out STATUS: UCL_RETURN);
```

Description / Parameters

Starts data acquisition on the local test node for the given enditem(s).

ITEM : individual enditem or group of enditems to be acquired.

ADU (optional) : the ADU Descriptor to be applied. If there is only one ADU Descriptor defined for ITEM, the parameter may be omitted. If more than one ADU Descriptor is defined and the parameter is omitted, the command starts acquisition for all ADUs . If ITEM is already of type ADU_DESCRIPTION, the parameter is ignored.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

START_ACQUISITION (\apm\dms\link\statistics, stat);

I-1.3.6.2 STOP_ACQUISITION

Specification

```
procedure  STOP_ACQUISITION
                    (in ITEM: ACQUISITION_COLLECTION;
                    in ADU: ADU_NAME := \\;
                    out  STATUS: UCL_RETURN);
```

Description / Parameters

Stops data acquisition on the local test node for the given enditems.

ITEM : individual enditem or group of enditems no longer to be acquired.

ADU (optional) : the ADU Descriptor to be applied. If only one ADU Descriptor is active for ITEM, the parameter may be omitted. If more than one ADU Descriptor is defined and the parameter is omitted, the command stops acquisition for all ADUs. If ITEM is already of type ADU_DESCRIPTION, the parameter is ignored.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

STOP_ACQUISITION (\apm\dms\link\statistics, stat);

I-1.3.6.3 SEND_SIMULATED_ADU

Specification

procedure SEND_SIMULATED_ADU
(in ADU: ADU_NAME;
out STATUS: UCL_RETURN);

Description / Parameters

Sends a specific ADU internally to the TES input buffers, in simulation mode. For example, triggers sending or simulation of a non–cyclic ADU.

ADU : name of simulated packet to be sent.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

SEND_SIMULATED_ADU (\apm\dms_scoe\ADU_3, stat);

I-1.3.6.4 WAIT_FOR_ADU

Specification

procedure WAIT_FOR_ADU
(in ADU: ADU_NAME;
out TIME_TAG: TIME;
out SEQUENCE_NUMBER: INTEGER;
out STATUS: UCL_RETURN);

Description / Parameters

Waits for an ADU to be supplied to this test node.

ADU : name of packet to be received.

TIME_TAG : returns the time tag from the received packet.

SEQUENCE_NUMBER : returns the current sequence number from the received packet.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

WAIT_FOR_ADU
(\apm\dms\housekeeping\packet, the_time_tag, the_sequence_number, stat);

I-1.3.6.5 SET_BITS_IN_SIMULATED_ADU

Specification

```
procedure  SET_BITS_IN_SIMULATED_ADU
                (in ADU: ADU_NAME;
                in OFFSET: INTEGER;
                in SIZE: BIT_COUNT;
                in VALUE: BITSET;
                out  STATUS: UCL_RETURN);
```

Description / Parameters

Modifies the bit–contents of a simulated ADU directly in simulation mode. The operation can only be applied on unstructured ADUs or telemetry ADUs.

ADU : name of the ADU to be modified.

OFFSET : location in bits in this ADU where bits are to be modified.

SIZE : the number of bits to modify. The upper limit is 32 bits.

VALUE : the new value of the set of bits to be modified.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
SET_BITS_IN_SIMULATED_ADU (\apm\dms\hk_packet_1, 1234, 3, {0,1,2},  stat);
```

I-1.3.6.6 SET_SIMULATED_ENDITEM_VALUE

Specification

```
PROCEDURE  SET_SIMULATED_ENDITEM_VALUE
                (in ITEM: MONITOR_ITEM_NAME;
                in BOOL_VALUE: BOOLEAN : =  FALSE;
                in INT_VALUE: INTEGER : =  0;
                in BITSET_VALUE: BITSET : =  {};
                in REAL_VALUE: REAL : =  0.0;
                in STR_VALUE: USER_MESSAGE : =  ””;
                out STATUS: UCL_RETURN);
```

Description / Parameters

Modifies the value of an enditem in simulated ADUs.
The operation can only be applied on measurements provided by structured ADUs.

ITEM : name of item to be modified.

Other parameters **:** To cope with the different data types a value can have, several parameters are provided. By selecting the appropriate parameter, eg INT_VALUE: 7, the desired value is set. The operation can only be applied on items provided by structured ADUs.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
SET_SIMULATED_ENDITEM_VALUE
                (\apm\dms\input_current, REAL_VALUE : 5.6, STATUS : stat);
```

I-1.3.6.7 ROUTE_TO_SAS

Specification

```
procedure  ROUTE_TO_SAS
        (in ITEM: ROUTE_SAS_ITEM_NAME   ;
        in SAS_NAME: APPLICATION_NAME;
        in OLD_SAS_NAME: APPLICATION_NAME :=  NULL_APPLICATION_NAME;
        out STATUS: UCL_RETURN);
```

Description / Parameters

Routes all items specified in ITEM which have an SAS Reference to the new SAS specified by SAS_NAME. ITEMs must be in state 'not acquired' if of type ADU_DESCRIPTION.

ITEM : The item to be re–defined in the local memory of the test node. ITEM can be

- a single enditem
- a incomplete pathname pointing to a virtual node, thus redefining all in that subtree of the types indicated in the type ROUTE_SAS_ITEM_NAME;
- a GDU_DESCRIPTION_LIST list, thus enabling all items in that list.

SAS_NAME: The new short name of the SAS to be applied for ITEM.
The SAS must run on the local test node .

OLD_SAS_NAME: The old short name of the SAS.Only items specified in ITEM having a SAS Reference matching this SAS name are changed .
If NULL_APPLICATION_NAME is given, all items specified in ITEM are changed.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

ROUTE_TO_SAS (\apm\cc_bus1\tc, ”VTC_SAS”, ”SSMB_SAS”, stat);

11.8.3.1.1 SET_CCSDS_APID

Specification

```
procedure SET_CCSDS_APID
          (in ITEM: CCSDS_ITEM_NAME ;
           in CCSDS_APID: INTEGER;
           in OLD_CCSDS_APID: INTEGER := –1;
           out STATUS: UCL_RETURN);
```

Description / Parameters

Routes all items specified in ITEM. ITEMs must be in state 'not acquired' if of type ADU_DESCRIPTION.

ITEM : The item to be re–defined in the local memory of the test node. ITEM can be

– a single enditem
– a incomplete pathname pointing to a virtual node, thus redefining all in that subtree of the types indicated in the type CCSDS_ITEM_NAME;
– a GDU_DESCRIPTION_LIST list, thus switching all items in that list.

CCSDS_APID: The new CCSDS_APID to be applied for the CCSDS Primary Header contained in ITEM.

OLD_CCSDS_APID: The old APID of the item.Only items specified in ITEM having an APID matching this OLD_CCSDS_APID are changed .
If a negative value is given, all items specified in ITEM are changed.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

SET_CCSDS_APID (\apm\cc_bus1\tc, 1234, 1235, stat);

I-1.3.7 Event Handling

I-1.3.7.1 LOG

Specification

procedure LOG
(in MESSAGE: USER_MESSAGE;
in LONG_TEXT: USER_MESSAGE;
out STATUS: UCL_RETURN)

Description / Parameters

Logs an event message to the test result database.
Logged events have group–code LOG and type–code MSG.

MESSAGE : string of up to 255 characters containing the message to be logged.

Note: If message is longer than 40 characters, and LONG_TEXT is empty, whole MESSAGE will be duplicated to appear as well in long text field of logged event .

LONG_TEXT : Additional message to be sent to user. Appears in the Long_Text field of the logged event.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

LOG (”Captain’s log, stardate 1 January 2137”,”No major events today”, stat);

I-1.3.7.2 USER_EVENT

Specification

procedure USER_EVENT
(in MESSAGE: USER_MESSAGE;
out STATUS: UCL_RETURN);

Description / Parameters

Logs an user event message to the test result database.
User events are special log event with the group–code UEVT and type–code MSG.

MESSAGE : string of up to 255 characters containing the message to be logged.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

USER_EVENT (”Begin_of_Test”, stat);

I-1.3.8 Engineering Value Logging

I-1.3.8.1 ENABLE_EVL

Specification

```
procedure  ENABLE_EVL
                    (in ITEM: MONITOR_COLLECTION;
                    out  STATUS: UCL_RETURN);
```

Description / Parameters

Enables engineering value logging (EVL).

ITEM : name of enditem(s) to be enabled.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
ENABLE_EVL  (\apm\power_scoe\current_sensors, stat);
```

I-1.3.8.2 DISABLE_EVL

Specification

```
procedure  DISABLE_EVL
                    (in ITEM: MONITOR_COLLECTION;
                    out STATUS: UCL_RETURN);
```

Description / Parameters

Disables engineering value logging. If the specified enditems are already disabled for EVL, has no effect.

ITEM : name of enditem(s) to be enabled.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
DISABLE_EVL  (\apm\power_scoe\current_sensors,  stat);
```

I-1.3.9 Archiving

I-1.3.9.1 ENABLE_ARCHIVING

Specification

procedure ENABLE_ARCHIVING
(in CYCLE: DURATION: = DEFAULT_ARCH_FILE_OPEN_PERIOD;
out STATUS: UCL_RETURN);

Description / Parameters

Enables archiving; creates a new archive file.

CYCLE : the time interval (1 minute to 24 hours exclusive) at which archive files are closed automatically. If archiving is already enabled and the cycle time of this call differs from the cycle time specified in the last call of this procedure, the current archive file is closed, and a new one is opened. Archiving periodically closes the archive file using the new cycle time.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

ENABLE_ARCHIVING(45 [min], stat);

I-1.3.9.2 DISABLE_ARCHIVING

Specification

procedure DISABLE_ARCHIVING;

Description / Parameters

Stops archiving on the local test node. If archiving was enabled the current archive file is closed.

Example Call

DISABLE_ARCHIVING;

I-1.3.9.3 CLOSE_ARCHIVE

Specification

procedure CLOSE_ARCHIVE
(out CLOSE_TIME : TIME;
out STATUS: UCL_RETURN);

Description / Parameters

Closes the currently open archive file and opens a fresh one, provided archiving is enabled.

CLOSE_TIME : returns the local time the archive file was closed.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

CLOSE_ARCHIVE (my_time_value, stat);

I-1.3.10 User Input & Output

I-1.3.10.1 READ_MESSAGE_FROM_USER

Specification

Specification

```
procedure READ_MESSAGE_FROM_USER
                (in PROMPT: USER_MESSAGE;
                in WORKSTATION: NODE_NAME: = \\;
                in OPTIONS: VALUE_STRING := ””;
                out USER_ENTRY: USER_MESSAGE;
                out STATUS: UCL_RETURN);
```

Description / Parameters

Reads a message from the user; issues a prompt and blocks until the user enters a message.

PROMPT : prompt string of up to 255 characters, on Workstation’s screen.

WORKSTATION (optional) : node to display the prompt; by default, the workstation that started this GTAP.

OPTIONS (optional): options specifying attributes for Pop–Up window

Syntax: –position X Y
–size W H
– foreground <colour>
– background <colour>

X,Y: coordinates; W: Width, H: Height
<colour>: colour name or hexadecimal specification

Example: ”–position 100 200 –size 600 150 –background red”

USER_ENTRY : returns what the user enters in a 255 character string, padded with NULLs.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

READ_MESSAGE_FROM_USER (”Continue?”, \apm\egse\test_ws\,””, buffer, stat);

I-1.3.10.2 WRITE_MESSAGE_TO_USER

Specification

```
procedure  WRITE_MESSAGE_TO_USER
                    (in MESSAGE: USER_MESSAGE;
                    in SUPPLEMENT: USER_MESSAGE := ””;
                    in MESSAGE_GROUP: MSG_GROUP := ””;
                    in MESSAGE_TYPE: MSG_TYPE := ”MSG ”;
                    in WORKSTATION: NODE_NAME:=\\;
                    in  ALL_WORKSTATIONS : BOOLEAN := FALSE;
                    out STATUS: UCL_RETURN);
```

Description / Parameters

Sends a message to the user of a given workstation.

MESSAGE : Message to be sent to user. Appears in the (short) text field of the message handler window. Is abbreviated to first 40 characters.
Note: If message is longer than 40 characters, and SUPPLEMENT is empty, whole MESSAGE will be duplicated to appear as well in supplement text field of message handler window .

SUPPLEMENT : Additional message to be sent to user. Appears in the supplement text field of the message handler window.

MSG_GROUP : A string of 4 character defining the message group. Might be one of the predefined CGS message groups, or defined by the user .

MSG_TYPE : A string of 4 character defining the message type. Message Types are mapped to the message class of the message handler.
The following are defined:

”INFO”:	mapped to ADVISORY
”MSG ”:	mapped to ADVISORY
”ERR ”:	mapped to ORDINARY
”WRN ”:	mapped to ADVISORY
”EXC ”:	mapped to SEVERE (to be used only for monitoring exceptions)
”ALRM”:	mapped to FATAL
any other:	mapped to ADVISORY

WORKSTATION (optional) : name of node to display message; by default the workstation where this GTAP (or its ancestors) was started.

ALL_WORKSTATIONS (optional) : if TRUE, MESSAGE is sent to all workstations connected to the test node. Parameter WORKSTATION is ignored in this case.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
WRITE_MESSAGE_TO_USER (”STEP 3”, , , , \apm\egse\test_ws, ,stat);
WRITE_MESSAGE_TO_USER (”SERIOUS ERROR”, ”NO DOWNLINK indicated anymore”,
                                        ”TM  ”, ”ALRM”, \apm\egse\test_ws, ,stat);
```

I-1.3.10.3 READ_NUMBER_FROM_USER

Specification

```
procedure READ_NUMBER_FROM_USER
                (in PROMPT_MESSAGE: USER_MESSAGE;
                 in WORKSTATION: NODE_NAME: = \\;
                 in OPTIONS: VALUE_STRING := ””;
                 out USER_ENTRY: REAL;
                 out STATUS: UCL_RETURN);
```

Description / Parameters

Reads a real number; issues a prompt and blocks until the user enters one at the workstation.

PROMPT : prompt string displayed on user's screen, up to 255 characters.

WORKSTATION (optional) : the workstation to use; by default that started this GTAP (or one of its ancestors).

OPTIONS (optional): options specifying attributes for Pop–Up window

Syntax: –position X Y
–size W H
– foreground <colour>
– background <colour>

X,Y: coordinates; W: Width, H: Height
<colour>: colour name or hexadecimal specification

Example: ”–position 100 200 –size 600 150 –foreground red”

USER_ENTRY : returns the number entered by the user.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

```
READ_NUMBER_FROM_USER
                (”Which Table ?”, \apm\egse\test_ws\, , my_number, stat);
```

I-1.3.11 Synoptic Display Control

I-1.3.11.1 DISPLAY_PICTURE

Specification

```
procedure  DISPLAY_PICTURE
        (in PICTURE: PICTURE_NAME;
        in WORKSTATION: NODE_NAME  := \\;
        in OPTIONS: VALUE_STRING := ””;
        out ID: PICTURE_ID;
        out STATUS: UCL_RETURN);
```

Description / Parameters

Displays a synoptic display on a specific workstation.

PICTURE : name of synoptic display.

WORKSTATION (optional): the work station to show the synoptic display: by default the work station from which the command originates, directly or via a GTAP.

OPTIONS (optional): options specifying attributes for window where picture is displayed

Syntax: –position X Y
–size W H
X,Y: coordinates; W: Width, H: Height   Example: ”–position 100 200 – size 600 150”

ID : returns identifier to be used by REMOVE_PICTURE when removing this synoptic display.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Example Call

```
DISPLAY_PICTURE
        (\apm\power\total_overview, \apm\egse\test_conductor\workstation,
        ”–position 100 200 –size 600 150”,  my_synoptic_ID, stat);
```

I-1.3.11.2 REMOVE_PICTURE

Specification

```
procedure  REMOVE_PICTURE
        (in ID: PICTURE_ID;
        in WORKSTATION: NODE_NAME  := \\;
        out STATUS: UCL_RETURN);
```

Description / Parameters

Remove a synoptic display from the screen of a specific workstation.

ID : the picture’s identifier, as returned from DISPLAY_PICTURE (section I-1.3.11.1).

WORKSTATION (optional): the work station showing the synoptic display: by default the work station from which the command originates, directly or via a GTAP.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Example Call

REMOVE_PICTURE (my_synoptic_ID, \apm\egse\test_conductor\workstation, stat);

I-1.3.12 UCL Code Reloading from MDB

I-1.3.12.1 LOAD_UCL

Specification

```
procedure  LOAD_UCL
                (in ITEM: UCL_ITEM_NAME;
                out STATUS: UCL_RETURN);
```

Description / Parameters

Loads the I–code of an AP or UCL User library into memory. This procedure can be used for two purposes: Either to preload the i–code to speed up the start–up of the AP/USER_LIB or to reload an AP/USER_LIB which has been recompiled in MDB and which has already been preloaded.

ITEM : the pathname of the item to be loaded.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Example Call

LOAD_UCL (\apm\egse\emergency_ap, stat);

I-1.3.13 Test Node Management

I-1.3.13.1 IS_LOCAL_NODE

Specification

function IS_LOCAL_NODE (in NODE : NODE_NAME) : BOOLEAN;

Description / Parameters

Returns TRUE if executed on the test node NODE, otherwise FALSE.

NODE : name of EGSE node

Example Call

BOOL := IS_LOCAL_NODE (\apm\egse\mtp);

11.8.3.1.2 SET_DEFAULT_WORKSTATION

Specification

```
procedure SET_DEFAULT_WORKSTATION
                        (in WORKSTATION: NODE_NAME := \\;
                        out STATUS: UCL_RETURN);
```

Description / Parameters

Sets the default workstation for a test node.
The default workstation(s) is/are used by all APs that are not started via interactive commands (e.g. APs started via monitoring exceptions/conditions).
By default (i.e. \\) the workstation from which the command originates, directly or via a GTAP.

WORKSTATION : node name indicating the workstation that shall become the default workstation.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

SET_DEFAULT_WORKSTATION (\apm\egse\main_ws, stat);

I-1.3.14 Conditions

I-1.3.14.1 ENABLE_CONDITIONS

Specification

```
procedure ENABLE_CONDITIONS
                        (in  ITEM: ACQUISITION_COLLECTION;
                         out STATUS: UCL_RETURN);
```

Description / Parameters

Enable condition for a specific measurement/software variable/derived value or for all enditems under a virtual tree/CDU or for all enditems within a monitor list or for all measurements within an ADU. Enables all conditions of the given enditems.
If acquisition is not enabled for ITEM, it will be automatically started

ITEM: The measurement(s) or software variable(s) carrying the conditions.

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : ITEM is not managed locally
- RUNTIME_ERROR : an unexpected error occurred

Example Call

```
ENABLE_CONDITIONS (\APM\mmu2, stat);
```

I-1.3.14.2 DISABLE_CONDITIONS

Specification

```
procedure DISABLE_CONDITIONS
                        (in  ITEM: ACQUISITION_COLLECTION;
                         out STATUS: UCL_RETURN);
```

Description / Parameters

Disable condition(s) for a specific measurement/software variable/derived value or for all enditems under a virtual tree/CDU or for all enditems within a monitor list or for all measurements within an ADU. Disables all conditions of the given enditems
When disabled, the conditions are still existing, and may be re–enabled via Enable_Conditions

ITEM: The measurement(s) or software variable(s) carrying the condition.

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : ITEM is not managed locally
- RUNTIME_ERROR : an unexpected error occurred

Example Call

```
DISABLE_CONDITIONS (\APM\mmu2, stat);
```

I-1.3.14.3 ENABLE_ON_INTEGER

Specification

```
procedure ENABLE_ON_INTEGER
          (in  ITEM             : ACQUISITION_COLLECTION;
           in  CONDITION_ITEM   : INTEGER_MONITOR_ITEM_NAME;
           in  CONDITION_CHECK  : CONDITION;
           in  CONDITION_VALUE  : INTEGER;
           in  SINGLE_SHOT      : BOOLEAN;
           out CONDITION_REF    : INTEGER;
           out STATUS           : UCL_RETURN);
```

Description / Parameters

Enables the processing of ITEM (measurement, software variable, monitoring list or measurements of a subtree) when the CONDITION_CHECK of the value of CONDITION_ITEM with CONDITION_VALUE is true (the condition is true).

If the condition is false, this will disable the processing of ITEM.By default, all enditems are enabled for processing. Conditions can be defined to disable and re–enable the processing of given end–items.

When the processing is enabled, ITEM will be authorised for calibration and monitoring. After ITEM has been enabled, it will be calibrated (if its acquisition has been started) and monitored (if its monitoring has been enabled) on reception of new values.

Enabling the processing of ITEM does not perform an ”initial” calibration / monitoring of ITEM in the case where it has already a value.

When the processing is disabled, ITEM will not be calibrated and not be monitored, even if its acquisition has been requested and if its monitoring is enabled.

CONDITION_ITEM must be maintained locally otherwise an error is generated. When the condition is triggered, only those measurements and SW variables identified by ITEM that are maintained locally will be processed.

It is not possible to set a condition where the ITEM to enable or disable is the CONDITION_ITEM. If ITEM is a group of enditems (i.e. monitoring list, ADU or incomplete pathname) containing CONDITION_ITEM, the condition will not apply for CONDITION_ITEM.

It is not possible to enable or disable the processing of a SW variable that is linked to an HK data.

ITEM : The item to be enabled, which can be

- a single enditem of type MEASUREMENT or SW variable;
- a incomplete pathname pointing to a virtual node, thus enabling all MEASUREMENTs or SW variables in that subtree;
- a monitor list, thus enabling all items in that list.

CONDITION_ITEM: The measurement or software variable carrying the condition
Only enditems having value type INTEGER are allowed

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : if true, the condition will be withdrawn after having been true.

CONDITION_REF : returns an unique identifier for the condition

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally
- INVALID_PARAMETER : ITEM and CONDITION_ITEM are identical
- or ITEM is a SW variable linked to an HK data.
- or CONDITION_CHECK is IN_RANGE (not supported)
- RUNTIME_ERROR : an unexpected error occurred

Example Call

ENABLE_ON_INTEGER (\apm\dms\mmu_2, \apm\dms\mmu_number, EQUAL, 2, false, c_ref, stat);

I-1.3.14.4 ENABLE_ON_FLOAT

Specification

```
procedure ENABLE_ON_FLOAT
        (in  ITEM           : ACQUISITION_COLLECTION;
         in  CONDITION_ITEM  : FLOAT_MONITOR_ITEM_NAME;
         in  CONDITION_CHECK : CONDITION;
         in  CONDITION_VALUE : REAL;
         in  SINGLE_SHOT     : BOOLEAN;
         out CONDITION_REF : INTEGER;
         out STATUS : UCL_RETURN);
```

Description / Parameters

Enables processing on conditions for float values;
Refer to analog description in ENABLE_ON_INTEGER

ITEM : Refer to analog description in ENABLE_ON_INTEGER;

CONDITION_ITEM: The measurement or software variable carrying the condition.
Only enditems having value type REAL are allowed

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : if true, the condition will be withdrawn after having been true.

CONDITION_REF : returns an unique identifier for the condition

STATUS : the return status of the operation

- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally
- INVALID_PARAMETER : ITEM and CONDITION_ITEM are identical
- or ITEM is a SW variable linked to an HK data.
- or CONDITION_CHECK is IN_RANGE (not supported)
- RUNTIME_ERROR : an unexpected error occurred

Example Call

ENABLE_ON_FLOAT (\apm\dms\fire, \apm\dms\smoke_sensor, GREATER, 100.0, false, c_ref, stat);

I-1.3.14.5 ENABLE_ON_STATECODE

Specification

```
procedure ENABLE_ON_STATECODE
        (in  ITEM            : ACQUISITION_COLLECTION;
         in  CONDITION_ITEM  : DISCRETE_MONITOR_ITEM_NAME;
         in  CONDITION_CHECK : DISCRETE_CONDITION;
         in  CONDITION_VALUE : STATECODE;
         in  SINGLE_SHOT     : BOOLEAN;
         out CONDITION_REF   : INTEGER;
         out STATUS          : UCL_RETURN);
```

Description / Parameters

Enables processing on conditions for discrete values;
Refer to analog description in ENABLE_ON_INTEGER

ITEM : Refer to analog description in ENABLE_ON_INTEGER;

CONDITION_ITEM: The measurement or software variable carrying the condition.
Only discrete enditems having value type STATECODE are allowed

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : if true, the condition will be withdrawn after having been true.

CONDITION_REF : returns an unique identifier for the condition

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally
- INVALID_PARAMETER : ITEM and CONDITION_ITEM are identical
- or ITEM is a SW variable linked to an HK data.
- RUNTIME_ERROR : an unexpected error occurred

Example Call

ENABLE_ON_STATECODE (\apm\dms\mmu_2, \apm\dms\mmu2_state, EQUAL, ON, false, c_ref, stat);

I-1.3.14.6 ENABLE_ON_BYTE_STREAM

Specification

```
procedure ENABLE_ON_BYTE_STREAM
        (in  ITEM             : ACQUISITION_COLLECTION;
         in  CONDITION_ITEM   : BYTE_STREAM_MONITOR_ITEM_NAME;
         in  CONDITION_CHECK  : DISCRETE_CONDITION;
         in  CONDITION_VALUE  : STRING;
         in  SINGLE_SHOT      : BOOLEAN;
         out CONDITION_REF    : INTEGER;
         out STATUS           : UCL_RETURN);
```

Description / Parameters

Enables processing on conditions for bytestream (string) values;
Refer to analog description in ENABLE_ON_INTEGER

ITEM : Refer to analog description in ENABLE_ON_INTEGER;

CONDITION_ITEM: The measurement or software variable carrying the condition.
Only enditems having raw value type BYTE_STREAM are allowed

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : if true, the condition will be withdrawn after having been true.

CONDITION_REF : returns an unique identifier for the condition

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally
- INVALID_PARAMETER : ITEM and CONDITION_ITEM are identical
- or ITEM is a SW variable linked to an HK data.
- RUNTIME_ERROR : an unexpected error occurred

Example Call

ENABLE_ON_BYTE_STREAM (\apm\dms\activation, \apm\dms\ap_name, equal, ”activation”, true, c_ref, stat);

I-1.3.14.7 SET_PROCESSING

Specification

```
procedure SET_PROCESSING
    (in  ITEM   : ACQUISITION_COLLECTION;
     in  SWITCH : ON_OFF;
     out STATUS : UCL_RETURN);
```

Description / Parameters

Enables or disables processing for enditems, overriding any conditions defined.

ITEM : Refer to analog description in ENABLE_ON_INTEGER;

SWITCH : ON to enable the processing and OFF to disable the processing.

STATUS : the return status of the operation

- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : ITEM is not managed locally
- INVALID_PARAMETER : ITEM is a SW variable linked to an HK data.
- RUNTIME_ERROR : an unexpected error occurred

Example Call

SET_PROCESSING (\apm\dms\mmu_2, OFF, stat);

I-1.3.14.8 GET_PROCESSING

Specification

```
procedure GET_PROCESSING_STATE
    (in  ITEM   : MONITOR_ITEM_NAME;
     out STATE  : ON_OFF;
     out STATUS : UCL_RETURN);
```

Description / Parameters

Fetches the status of processing (enabled, disabled) for an enditem

ITEM : A single measurement or software variable

STATE : ON if enabled for processing and OFF if disabled for processing.

STATUS : the return status of the operation

- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : ITEM is not managed locally
- RUNTIME_ERROR : an unexpected error occurred

Example Call

GET_PROCESSING_STATE (\apm\dms\mmu_2, proc_state, stat);

I-1.3.14.9 SET_LIMIT_SET_ON_INTEGER

Specification

```
procedure SET_LIMIT_SET_ON_INTEGER
          (in  ITEM            : MONITOR_COLLECTION;
           in  LIMIT_SET       : LIMIT_SET_NUMBER;
           in  CONDITION_ITEM  : INTEGER_MONITOR_ITEM_NAME;
           in  CONDITION_CHECK : CONDITION;
           in  CONDITION_VALUE : INTEGER;
           in  SINGLE_SHOT     : BOOLEAN;
           out CONDITION_REF   : INTEGER;
           out STATUS          : UCL_RETURN);
```

Description / Parameters

Sets the LIMIT_SET for ITEM (measurement or software variable) when the CONDITION_CHECK of the value of CONDITION_ITEM with CONDITION_VALUE is true (the condition is true).

This will not perform an ”initial” monitoring of ITEM in the case where it has already a value. The monitoring will be performed on reception of the next value (in case the monitoring has been enabled).
This can lead to a temporary ”inconsistent” output of the procedure GET_FULL_ENDITEM_MONITOR_STATUS (result of the monitoring with the old limit set with values of the new limit set might be inconsistent).

ITEM and CONDITION_ITEM must be maintained locally on the same node otherwise an error is generated.

It is not possible to set a condition where the ITEM is identical to the CONDITION_ITEM.

The LIMIT_SET must be defined for ITEM, otherwise an error is generated

In case a set of items is addressed, the limit set is switched to LIMIT_SET only, if the limit set is defined. No error message is given for undefined sets.

ITEM : Measurement(s), derived value(s) or software variable(s).

LIMIT_SET : The limit set number to select.
The LIMIT_SET must be defined for ITEM, otherwise an error is generated

CONDITION_ITEM: The measurement or software variable/derived values carrying the condition.
Only enditems having value type INTEGER are allowed

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : if true, the condition will be withdrawn after having been true.

CONDITION_REF : returns an unique identifier for the condition

STATUS : the return status of the operation
— OK : operation successfull
— INVALID_TESTNODE_MODE : the test node is not in executing mode
— INVALID_ITEM_NAME : ITEM or CONDITION_ITEM are not managed locally
— INVALID_PARAMETER : ITEM and CONDITION_ITEM are identical
— or CONDITION_CHECK is IN_RANGE (not supported)

— INVALID_LIMIT_SET : the limit set is not defined for ITEM
— RUNTIME_ERROR : an unexpected error occurred

Example Call
SET_LIMIT_SET_ON_INTEGER (\apm\dms\mmu_status,5, \apm\dms\mmu, equal, 2, true, c_ref, stat);

I-1.3.14.10 SET_LIMIT_SET_ON_FLOAT

Specification
```
procedure SET_LIMIT_SET_ON_FLOAT
          (in  ITEM            : MONITOR_COLLECTION;
           in  LIMIT_SET       : LIMIT_SET_NUMBER;
           in  CONDITION_ITEM  : FLOAT_MONITOR_ITEM_NAME;
           in  CONDITION_CHECK : CONDITION;
           in  CONDITION_VALUE : REAL;
           in  SINGLE_SHOT     : BOOLEAN;
           out CONDITION_REF   : INTEGER;
           out STATUS          : UCL_RETURN);
```

Description / Parameters
Set a limit set on conditions for float values;
Refer to analog description in SET_LIMIT_SET_ON_INTEGER

ITEM : Measurement(s), derived value(s) or software variable(s).

LIMIT_SET : The limit set number to select.
The LIMIT_SET must be defined for ITEM, otherwise an error is generated

CONDITION_ITEM: The measurement or software variable/derived values carrying the condition.
Only enditems having value type REAL are allowed

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : if true, the condition will be withdrawn after having been true.

CONDITION_REF : returns an unique identifier for the condition

STATUS : the return status of the operation
— OK : operation successfull
— INVALID_TESTNODE_MODE : the test node is not in executing mode
— INVALID_ITEM_NAME : ITEM or CONDITION_ITEM are not managed locally
— INVALID_PARAMETER : ITEM and CONDITION_ITEM are identical
— INVALID_LIMIT_SET : the limit set is not defined for ITEM
— or CONDITION_CHECK is IN_RANGE (not supported)
— RUNTIME_ERROR : an unexpected error occurred

Example Call
SET_LIMIT_SET_ON_FLOAT (\apm\dms\temp,5, \apm\dms\temp, greater, 25.0, true, c_ref, stat);

I-1.3.14.11 SET_LIMIT_SET_ON_STATECODE

Specification

```
procedure SET_LIMIT_SET_ON_STATECODE
        (in  ITEM            : MONITOR_COLLECTION;
         in  LIMIT_SET       : LIMIT_SET_NUMBER;
         in  CONDITION_ITEM  : DISCRETE_MONITOR_ITEM_NAME;
         in  CONDITION_CHECK : DISCRETE_CONDITION;
         in  CONDITION_VALUE : STATECODE;
         in  SINGLE_SHOT     : BOOLEAN;
         out CONDITION_REF   : INTEGER;
         out STATUS          : UCL_RETURN);
```

Description / Parameters

Set a limit set on conditions for statecode values;
Refer to analog description in SET_LIMIT_SET_ON_INTEGER

ITEM : A single measurement or software variable/derived value, a monitor list or a virtual path.

LIMIT_SET : The limit set number to select.
The LIMIT_SET must be defined for ITEM, otherwise an error is generated

CONDITION_ITEM: The measurement or software variable/derived value carrying the condition.
Only discrete enditems having value type STATECODE are allowed

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : if true, the condition will be withdrawn after having been true.

CONDITION_REF : returns an unique identifier for the condition

STATUS : the return status of the operation

- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : ITEM or CONDITION_ITEM are not managed locally
- INVALID_PARAMETER : ITEM and CONDITION_ITEM are identical
- INVALID_LIMIT_SET : the limit set is not defined for ITEM
- RUNTIME_ERROR : an unexpected error occurred

Example Call

SET_LIMIT_SET_ON_STATECODE (\apm\dms\temp,5, \apm\dms\status, equal, ”heating”, true, c_ref, stat);

I-1.3.14.12 SET_LIMIT_SET_ON_BYTESTREAM

Specification

procedure SET_LIMIT_SET_ON_BYTE_STREAM
(in ITEM : MONITOR_COLLECTION;
in LIMIT_SET : LIMIT_SET_NUMBER;
in CONDITION_ITEM : BYTE_STREAM_MONITOR_ITEM_NAME;
in CONDITION_CHECK : DISCRETE_CONDITION;
in CONDITION_VALUE : STRING;
in SINGLE_SHOT : BOOLEAN;
out CONDITION_REF : INTEGER;
out STATUS : UCL_RETURN);

Description / Parameters

Set a limit set on conditions for byte stream values;
Refer to analog description in SET_LIMIT_SET_ON_INTEGER

ITEM : A single measurement or software variable/derived value, a monitor list or a virtual path.

LIMIT_SET : The limit set number to select.
The LIMIT_SET must be defined for ITEM, otherwise an error is generated

CONDITION_ITEM: The measurement or software variable/derived value carrying the condition.
Only enditems having raw value type BYTE_STREAM
(engineering value type STRING) are allowed

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : if true, the condition will be withdrawn after having been true.

CONDITION_REF : returns an unique identifier for the condition

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : ITEM or CONDITION_ITEM are not managed locally
- INVALID_PARAMETER : ITEM and CONDITION_ITEM are identical
- INVALID_LIMIT_SET : the limit set is not defined for ITEM
- RUNTIME_ERROR : an unexpected error occurred

Example Call

SET_LIMIT_SET_ON_BYTE_STREAM (\apm\dms\temp,2, \apm\dms\ap_name, equal, ”deactiv”, true, c_ref, stat);

I-1.3.14.13 START_AP_ON_INTEGER

Specification

```
procedure START_AP_ON_INTEGER
    (in  AP              : AP_NAME;
     in  CONDITION_ITEM  : INTEGER_MONITOR_ITEM_NAME;
     in  CONDITION_CHECK : CONDITION;
     in  CONDITION_VALUE : INTEGER;
     in  SINGLE_SHOT     : BOOLEAN;
     out CONDITION_REF   : INTEGER;
     out STATUS          : UCL_RETURN);
```

Description / Parameters

Starts an automated procedure when the CONDITION_CHECK of the value of CONDITION_ITEM with CONDITION_VALUE is true (the condition is true).

CONDITION_ITEM must be maintained locally on the node otherwise an error is generated.

The AP to start must be an AP without parameters.

AP : the name of the automated procedure to start. This must be an AP without parameter or where all parameters have default values.

CONDITION_ITEM: The measurement or software variable carrying the condition
Only enditems having value type INTEGER are allowed

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : if true, the condition will be withdrawn after having been true.

CONDITION_REF : returns an unique identifier for the condition

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally
- INVALID_PARAMETER : CONDITION_CHECK is IN_RANGE (not supported)
- RUN-TIME_ERROR : an unexpected error occurred

Example Call

START_AP_ON_INTEGER (\announce_mmu_2, \apm\dms\mmu_number, EQUAL, 2, true, c_ref, stat);

I-1.3.14.14 START_AP_ON_FLOAT

Specification

```
procedure START_AP_ON_INTEGER
          (in  AP              : AP_NAME;
           in  CONDITION_ITEM  : FLOAT_MONITOR_ITEM_NAME;
           in  CONDITION_CHECK : CONDITION;
           in  CONDITION_VALUE : REAL;
           in  SINGLE_SHOT     : BOOLEAN;
           out CONDITION_REF   : INTEGER;
           out STATUS          : UCL_RETURN);
```

Description / Parameters

Starts an AP on conditions for float values;
Refer to analog description in START_AP_ON_INTEGER

AP : the name of the automated procedure to start. This must be an AP without parameter or where all parameters have default values.

CONDITION_ITEM: The measurement or software variable carrying the condition
Only enditems having value type REAL are allowed

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : if true, the condition will be withdrawn after having been true.

CONDITION_REF : returns an unique identifier for the condition

STATUS : the return status of the operation
— OK : operation successfull
— INVALID_TESTNODE_MODE : the test node is not in executing mode
— INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally
— INVALID_PARAMETER : CONDITION_CHECK is IN_RANGE (not supported) — RUN-TIME_ERROR : an unexpected error occurred

Example Call

START_AP_ON_FLOAT (\switchoff_mmu2, \apm\dms\mmu_temp, GREATER, 35.0, false, c_ref, stat);

I-1.3.14.15 START_AP_ON_STATECODE

Specification

```
procedure START_AP_ON_STATECODE
    (in  AP              : AP_NAME;
     in  CONDITION_ITEM  : DISCRETE_MONITOR_ITEM_NAME;
     in  CONDITION_CHECK : DISCRETE_CONDITION;
     in  CONDITION_VALUE : STATECODE;
     in  SINGLE_SHOT     : BOOLEAN;
     out CONDITION_REF   : INTEGER;
     out STATUS          : UCL_RETURN);
```

Description / Parameters

Starts an AP on conditions for statecode values;
Refer to analog description in START_AP_ON_INTEGER

AP : the name of the automated procedure to start. This must be an AP without parameter or where all parameters have default values.

CONDITION_ITEM: The measurement or software variable carrying the condition
Only enditems having value type STATECODE are allowed

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : if true, the condition will be withdrawn after having been true.

CONDITION_REF : returns an unique identifier for the condition

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally
- RUNTIME_ERROR : an unexpected error occurred

Example Call

```
START_AP_ON_STATECODE (\switchoff_mmu2, \apm\dms\power_stat,  EQUAL, ”LOW”, false, c_ref, stat);
```

I-1.3.14.16 START_AP_ON_BYTE_STREAM

Specification

```
procedure START_AP_ON_BYTE_STREAM
        (in  AP              : AP_NAME;
         in  CONDITION_ITEM  : BYTE_STREAM_MONITOR_ITEM_NAME;
         in  CONDITION_CHECK : DISCRETE_CONDITION;
         in  CONDITION_VALUE : STRING;
         in  SINGLE_SHOT     : BOOLEAN;
         out CONDITION_REF   : INTEGER;
         out STATUS          : UCL_RETURN);
```

Description / Parameters

Starts an AP on conditions for byte stream values;
Refer to analog description in START_AP_ON_INTEGER

AP : the name of the automated procedure to start. This must be an AP without parameter or where all parameters have default values.

CONDITION_ITEM: The measurement or software variable carrying the condition Only enditems having value type STRING (raw type: BYTE_STREAM) are allowed

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : if true, the condition will be withdrawn after having been true.

CONDITION_REF : returns an unique identifier for the condition

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally
- RUNTIME_ERROR : an unexpected error occurred

Example Call

START_AP_ON_BYTE_STREAM (\switchoff_mmu2, \apm\dms\mmu_use, EQUAL, ”is not used”, false, c_ref, stat);

I-1.3.14.17 WITHDRAW_CONDITION

Specification

```
procedure WITHDRAW_CONDITION
        (in  CONDITION_ITEM : MONITOR_ITEM_NAME;
         in  CONDITION_REF  : INTEGER := ALL_CONDITIONS;
         out STATUS         : UCL_RETURN);
```

Description / Parameters

Removes a condition or all conditions of a CONDITION_ITEM.

CONDITION_ITEM: The measurement or software variable carrying the condition

CONDITION_REF : The unique identifier for the condition. If ALL_CONDITIONS is given, all conditions defined for CONDITION_ITEM are removed.

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally or
- CONDITION_REF does not identify an existing
- condition
- RUNTIME_ERROR : an unexpected error occurred

Example Call

```
WITHDRAW_CONDITION (\apm\dms\mmu_use,  125, stat);
```

I-1.3.14.18 WITHDRAW_ALL_CONDITIONS

Specification

```
procedure WITHDRAW_ALL_CONDITIONS
        (out STATUS         : UCL_RETURN);
```

Description / Parameters

Removes all conditions defined in the local test node.

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- RUNTIME_ERROR : an unexpected error occurred

Example Call

```
WITHDRAW_ALL_CONDITIONS (stat);
```

I-1.3.14.19 NUMBER_OF_CONDITION_ITEMS

Specification

```
function NUMBER_OF_CONDITION_ITEMS : INTEGER;
```

Description / Parameters

Returns the number of items carrying conditions.

Example Call

```
NB_CONDITIONS := NUMBER_OF_CONDITION_ITEMS;
```

I-1.3.14.20 GET_CONDITION_ITEM

Specification

```
procedure GET_CONDITION_ITEM
      (in  ITEM_NUMBER    : INTEGER;
       out CONDITION_ITEM : MONITOR_ITEM_NAME;
       out STATUS         : UCL_RETURN);
```

Description / Parameters

Gets the name of the CONDITION_ITEM identified by an index.
To get all the CONDITION_ITEM's, this procedure has to be called with all numbers between 1 and the value of NUMBER_OF_CONDITION_ITEMS.

ITEM_NUMBER : index of the CONDITION_ITEM

CONDITION_ITEM : the measurement or software variable carrying the condition.

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_PARAMETER : There is no CONDITION_ITEM at the index ITEM_NUMBER
- RUNTIME_ERROR : an unexpected error occurred

Example Call

```
for ITEM_NUMBER := 1 to NUMBER_OF_CONDITION_ITEMS do
        GET_CONDITION_ITEM(ITEM_NUMBER, PATHNAME_ARRAY(ITEM_NUMBER),STAT);
end for;
```

I-1.3.14.21 NUMBER_CONDITIONS

Specification

```
function NUMBER_CONDITIONS
    (in  CONDITION_ITEM : MONITOR_ITEM_NAME): INTEGER;
```

Description / Parameters

Returns the number of conditions carried by a CONDITION_ITEM. if CONDITION_ITEM is not maintained locally, 0 is returned

Example Call

```
NB_CONDITIONS := NUMBER_CONDITIONS(\apm\dms\mmu_2_status);
```

I-1.3.14.22 GET_CONDITION

Specification

```
procedure GET_CONDITION
    (in  CONDITION_ITEM   : MONITOR_ITEM_NAME;
     in  CONDITION_NUMBER : INTEGER;
     out CONDITION_REF    : INTEGER;
     out CONDITION_CHECK  : CONDITION;
     out CONDITION_VALUE  : CONDITION_VAL;
     out SINGLE_SHOT      : BOOLEAN;
     out STATE            : CONDITION_STATE;
     out ACTION           : ACTION_DESCRIPTION;
     out STATUS           : UCL_RETURN);
```

Description / Parameters

Provide the description of a condition.

CONDITION_ITEM : the name of the item carrying the condition.

CONDITION_NUMBER : the index of the condition (between 1 and the value of GET_NUMBER_CONDITIONS).

CONDITION_REF : the unique identifier of the condition.

CONDITION_CHECK : the check to apply on CONDITION_ITEM.

CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.

SINGLE_SHOT : indicates if the condition is to be removed after having been true.

STATE : indicates the state of the condition. It can be
- IS_TRUE : the condition is true, the associated action has been triggered
- IS_FALSE : the condition is false, the associated action has been triggered (only for the enable/disable conditions)
- IS_UNKNOWN : the condition has been set but has not yet been analysed (CONDITION_ITEM has not a valid value)

ACTION : provide the description of the condition, i.e:
- the sid of the item(s) for which the processing is to be enabled
- the sid of the item for which a new limit set is to be set and the limit set number
- the sid of an AP

STATUS : the return status of the operation
- OK : operation successfull
- INVALID_TESTNODE_MODE : the test node is not in executing mode
- INVALID_PARAMETER : There is no (more) condition at the index
- CONDITION_NUMBER (see above note).
- INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally.
- RUNTIME_ERROR : an unexpected error occurred

Example Call

```
GET_CONDITION(\apm\dms\mmu_stat,2,c_ref,c_check,c_value,single,action,stat);
```

I-1.3.15 General Conversion Routines

I-1.3.15.1 PATHNAME_TO_STRING

Specification

```
procedure PATHNAME_TO_STRING
                        (in PATH : PATHNAME;
                         out NAME: STRING);
```

Description / Parameters

Convert the pathname type into a string type

PATH : the pathname to be converted.

NAME : the pathname returned as a string
If PATH is not existing, an empty string is returned
If NAME is too short to hold resulting string, the string is truncated

Example Call

```
VARIABLE NAME_STRING : STRING(255);
VARIABLE STATUS: INTEGER;
PATHNAME_TO_STRING(\EURECA\EGSE\TEST\LIBS\SAS_ITEM,  NAME_STRING);
WRITE_MESSAGE_TO_APPLICATION (1234,  NAME_STRING,     STATUS);
```

I-1.3.15.2 PATH

Specification

```
function PATH(in NAME: STRING) : PATHNAME;
```

Description / Parameters

Convert the string type into a pathname type

NAME : the pathname as a string
returns Null–Pathname in case the path is not existing

Example Call

```
VARIABLE NAME_STRING : STRING(255);
VARIABLE SAS_PATH: PATHNAME;
VARIABLE STATUS: INTEGER;

READ_MESSAGE_FROM_APPLICATION (TRUE, 1234,  NAME_STRING,     STATUS);
SAS_PATH := PATH( NAME_STRING);
```

I-1.3.15.3 STATE_CODE_TO_STRING

Specification

procedure STATECODE_TO_STRING (in CODE : STATECODE;
out CODE_TEXT: STRING);

Description / Parameters

Convert the statecode type into a string type
CODE_TEXT should have a length of 8
If CODE_TEXT is too short to hold resulting string, the string is truncated
If CODE_TEXT is longer than the statecode, it is filled with blanks

CODE : the statecode
returns Null–Pathname in case the path is not existing

CODE_TEXT : the statecode as a string (without ”$” as prefix)

Example Call

VARIABLE CODE_STRING : STRING(8);
STATECODE_TO_STRING($OFF, CODE_STRING);

I-1.3.15.4 CODE

Specification

function CODE(in CODE_TEXT : STRING) : STATECODE;

Description / Parameters

Convert the string type into a statecode type
If CODE_TEXT is longer than 8 characters, the string is truncated.
If CODE_TEXT is invalid in term of a STATECODE, the statecode returned is either an ”empty” startecode (i.e. has no characters) if the first character is invalid or a statecode corresponding to the CODE_TEXT truncated at the first illegal character.
If CODE_TEXT is shorter than 8 characters, the STATECODE is extended by blanks

CODE_TEXT : the statecode (without ”$”) as a string;

Example Call

VARIABLE S_CODE : STATECODE;
S_CODE := CODE(”OFF”);

I-2 UCL Ground_Commands_To_Onboard System Library

This library contains commands sent to the COF DMS system.

Library Id (Body_Id): 4

I-2.1 Routines Summary

DISABLE_SW_COMMAND
ENABLE_SW_COMMAND
EXECUTE_FLAP
EXECWAIT_FLAP
ISSUE_SW_COMMAND
ROUTE_SWOP_TO_SAS
SET_CCSDS_END_POINT
SET_DEVICE_ADDRESS
GET_CCSDS_PACKET_TYPE
SET_CCSDS_PACKET_TYPE

I-2.2 UCL Ground_Commands_to_Onboard System Library Specification

```
------------------------------------------------------------------------
--
--*****************************************
-- GROUND_TO_OB_LIB System Library
--*****************************************
--ABSTRACT--
--     Defines Procedures to send commands to the onboard system
--     Must be compiled with body ID = 4
--
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  : GROUND_TO_OB_LIB System Library
--*---------------------------------------------------------------------
--CONTENTS--
--      COMPILER     : UCLC
--      LANGUAGE     : UCL
--CHANGE HISTORY
--      PIRN 8101 1/A: provide max_data_length and received_data_length for
--                     issue_sw_command; new UCL_RETURN RESPONSE_TOO_SHORT
--                     (SPRs 7288. Cleanup of inconsistencies in return codes in
--                     ISSUE_SW_COMMAND, EXECUTE_FLAP and EXECWAIT_FLAP.
--                     new UCL_RETURN RESPONSE_TIMEOUT.
--                     add operations to set/get the packet type.
--                 change library name to GROUND_TO_OB_LIB (SPR-7826)
--      PIRN 8011 1/A: provide operation ROUTE_SWOP_TO SAS to set the SAS for an
--                     SW Command or the response packet.
--      PIRN 8011 1/-: provide operation to set the DEVICE_ADDRESS for an
--                     APPLICATION_ID identified by the source and destination.
--      PIRN 7004 for CGS V4.1.0: Add ENABLE/DISABLE_SW_COMMAND
--                                Add Ground_Node parameter to
--                                ISSUE_SW_COMMAND and EXEC..._FLAP
--                                Add new return stati
--                                Add onboard_execution node, onb_priority,
--                                time_tag
--      PIRN 6012 for CGS V4.0.0: Add SET_CCSDS_END_POINT /
--                                update cmds
--END HISTORY
--
------------------------------------------------------------------------

library GROUND_TO_OB_LIB;


------------------------------------------------------------------------
-- CONSTANTS AND TYPES
------------------------------------------------------------------------

type END_POINT  = PATHNAME (CCSDS_END_POINT);
```

```
type FLAP_NAME  = PATHNAME (UCL_AUTOMATED_PROCEDURE);
type SW_COMMAND = PATHNAME (SWOP_COMMAND);

type SW_COMMANDS_AND_RESPONSES = PATHNAME (SWOP_COMMAND,
                                           RESPONSE_PACKET,
                                           VIRTUAL);



--
-- Type and Constants defining priority
--

type PRIORITY = (LOW, HIGH);

constant ONBOARD_LOW:                     INTEGER := 1;
constant ONBOARD_HIGH:                    INTEGER := 2;
constant DEFAULT_ONBOARD_PRIORITY:        INTEGER := ONBOARD_LOW;

type ONBOARD_PRIORITY = INTEGER (ONBOARD_LOW .. ONBOARD_HIGH);



--
-- Constants defining timeouts
--

constant DEFAULT_SW_COMMAND_TIMEOUT: DURATION := 10.0 [s];
constant DEFAULT_FLAP_TIMEOUT:       DURATION := 30.0 [s];



--
-- Type and Constants defining the status returned for UCL procedures
--
type UCL_RETURN     = INTEGER;

constant OK:                           UCL_RETURN :=   1;
constant BUSY:                         UCL_RETURN :=   2;
constant INVALID_ITEM_NAME:            UCL_RETURN :=   9;
constant INVALID_APPLICATION_ID:       UCL_RETURN :=  12;
constant INVALID_NODE_NAME:            UCL_RETURN :=  13;
constant ITEM_IS_DISABLED:             UCL_RETURN :=  18;
constant MESSAGE_TOO_BIG:              UCL_RETURN :=  20;
constant NO_ADU_SERVICE:               UCL_RETURN :=  21;
constant APPLICATION_NACK:             UCL_RETURN :=  23;
constant INVALID_APPLICATION_NAME:     UCL_RETURN :=  24;
constant APPLICATION_NOT_READY:        UCL_RETURN :=  25;
constant TIMEOUT:                      UCL_RETURN :=  26;
constant INVALID_ADU_TYPE:             UCL_RETURN :=  30;
constant INVALID_TESTNODE_MODE:        UCL_RETURN :=  31;
constant ITEM_NOT_SAS_COMPATIBLE:      UCL_RETURN :=  32;
```

```
constant NO_GDU_SERVICE:              UCL_RETURN :=  35;
constant INVALID_TIME:                UCL_RETURN :=  37;
constant OPERATION_ABORTED:           UCL_RETURN :=  39;
constant PROTOCOL_ERROR:              UCL_RETURN :=  48;
constant PARAMETER_ERROR:             UCL_RETURN :=  49;
constant COMMUNICATION_ERROR:         UCL_RETURN :=  54;
constant INVALID_NODE_TYPE:           UCL_RETURN :=  56;
constant SW_COMMAND_NOT_FOUND:        UCL_RETURN :=  58;
constant RESPONSE_TOO_SHORT:          UCL_RETURN :=  59;
constant RESPONSE_TIMEOUT:            UCL_RETURN :=  60;
constant RUNTIME_ERROR:               UCL_RETURN := 100;

--
-- Types and Constants defining the status returned in the response packet
--

type FLAP_RETURN = INTEGER;
type ONBOARD_RETURN = INTEGER;
type FLAP_ID  = INTEGER;

constant ONB_RESULT_SUCCESS:                   ONBOARD_RETURN :=  0;
constant ONB_RESULT_SEQUENCE_COUNT_ANOMALY:    ONBOARD_RETURN :=  1;
constant ONB_RESULT_NOT_AUTHORIZED:            ONBOARD_RETURN :=  2;
constant ONB_RESULT_UNKNOWN_COMMAND:           ONBOARD_RETURN :=  3;
constant ONB_RESULT_COMMAND_NOT_VALID:         ONBOARD_RETURN :=  4;
constant ONB_RESULT_PARAMETERS_INCOMPLETE:     ONBOARD_RETURN :=  5;
constant ONB_RESULT_PARAMETERS_NOT_VALID:      ONBOARD_RETURN :=  6;
constant ONB_RESULT_SEQUENCE_COUNT_REPETITION: ONBOARD_RETURN :=  7;
constant ONB_RESULT_SEQUENCE_COUNT_MISMATCH:   ONBOARD_RETURN :=  8;
constant ONB_RESULT_CHECKSUM_ERROR:            ONBOARD_RETURN :=  9;
constant ONB_RESULT_TWO_STAGE_BUF_FULL:        ONBOARD_RETURN :=  10;
constant ONB_RESULT_TWO_STAGE_SOURCE_MISMATCH: ONBOARD_RETURN :=  11;
constant ONB_RESULT_OTHER:                     ONBOARD_RETURN :=  63;

--
-- Type for setting the device address linked to an application id
--

type DEVICE_ADDRESS = STRING (20);

--
-- Type for setting the SAS name defined for SWOP and Response packets
--

type  SAS_NAME = STRING (20);

constant NULL_SAS_NAME : SAS_NAME := "";
```

```
constant NO_DATA_LENGTH_LIMIT : INTEGER := 4081;
–– maximum length of data field + 1;

type DATA_LENGTH_RANGE = INTEGER (0 .. NO_DATA_LENGTH_LIMIT);


––
–– type for setting the packet type of a swop command
––

constant CCSDS_DEFAULT_PACKET           : INTEGER :=  0;
constant CCSDS_MEMORY_DUMP_PACKET       : INTEGER :=  1;
constant CCSDS_DATA_SEGMENT_PACKET      : INTEGER :=  2;
constant CCSDS_ESSENTIAL_HK_PACKET      : INTEGER :=  3;
constant CCSDS_SYSTEM_HK_PACKET         : INTEGER :=  4;
constant CCSDS_PAYLOAD_HK_PACKET        : INTEGER :=  5;
constant CCSDS_SCIENCE_PACKET           : INTEGER :=  6;
constant CCSDS_SSMB_ANCILLIARY_PACKET   : INTEGER :=  7;
constant CCSDS_ESSENTIAL_COMMAND_PACKET : INTEGER :=  8;
constant CCSDS_SYSTEM_COMMAND_PACKET    : INTEGER :=  9;
constant CCSDS_PAYLOAD_COMMAND_PACKET   : INTEGER := 10;
constant CCSDS_MEMORY_LOAD_PACKET       : INTEGER := 11;
constant CCSDS_RESPONSE_PACKET          : INTEGER := 12;
constant CCSDS_REPORT_PACKET            : INTEGER := 13;
constant CCSDS_EXCEPTION_PACKET         : INTEGER := 14;
constant CCSDS_ACKNOWLEDGE_PACKET       : INTEGER := 15;

type CCSDS_PACKET_TYPE
       = INTEGER (CCSDS_DEFAULT_PACKET .. CCSDS_ACKNOWLEDGE_PACKET);


–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––
–– Onboard Software Commanding / Flight Application Control
–––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

procedure SET_CCSDS_END_POINT
            (in  GROUND_NODE : END_POINT;
             out STATUS      : UCL_RETURN);

procedure ISSUE_SW_COMMAND
            (in  SW_CMD          : SW_COMMAND ();
             in  ONBOARD_NODE    : END_POINT;
             in  GROUND_NODE     : END_POINT := \\;
             in  PRIO            : PRIORITY := LOW;
             in  TIMEOUT         : DURATION := default_sw_command_timeout;
             in  MAX_DATA_LENGTH : DATA_LENGTH_RANGE := NO_DATA_LENGTH_LIMIT;
             out RECEIVED_DATA_LENGTH : DATA_LENGTH_RANGE;
             out TRANSACTION_ID  : INTEGER;
```

```
             out RESULT          : ONBOARD_RETURN;
             out STATUS          : UCL_RETURN);

procedure ENABLE_SW_COMMAND
            (in  SW_CMD : SW_COMMAND;
             out STATUS : UCL_RETURN);

procedure DISABLE_SW_COMMAND
            (in  SW_CMD : SW_COMMAND;
             out STATUS : UCL_RETURN);

procedure EXECUTE_FLAP
             (in  FLAP                   : FLAP_NAME ();
              in  ONBOARD_RECEPTION_NODE : END_POINT;
              in  ONBOARD_EXECUTION_NODE : END_POINT := \\;
              in  GROUND_NODE            : END_POINT := \\;
              in  GROUND_PRIO            : PRIORITY := LOW;
              in  ONBOARD_PRIO : ONBOARD_PRIORITY := default_onboard_priority;
              in  TIME_TAG               : TIME    := ~:~;
              in  TIMEOUT                : DURATION := default_flap_timeout;
              out TRANSACTION_ID         : INTEGER;
              out RESULT                 : ONBOARD_RETURN;
              out ID                     : FLAP_ID;
              out FLAP_EXEC_STATUS       : FLAP_RETURN;
              out STATUS                 : UCL_RETURN);

procedure EXECWAIT_FLAP
             (in  FLAP                   : FLAP_NAME ();
              in  ONBOARD_RECEPTION_NODE : END_POINT;
              in  ONBOARD_EXECUTION_NODE : END_POINT := \\;
              in  GROUND_NODE            : END_POINT := \\;
              in  GROUND_PRIO            : PRIORITY := LOW;
              in  ONBOARD_PRIO : ONBOARD_PRIORITY := default_onboard_priority;
              in  TIME_TAG               : TIME    := ~:~;
              in  TIMEOUT                : DURATION := default_flap_timeout;
              out TRANSACTION_ID         : INTEGER;
              out RESULT                 : ONBOARD_RETURN;
              out ID                     : FLAP_ID;
              out FLAP_EXEC_STATUS       : FLAP_RETURN;
              out STATUS                 : UCL_RETURN);

procedure SET_DEVICE_ADDRESS
            (in SOURCE      : END_POINT;
             in DESTINATION : END_POINT;
             in ADDRESS     : DEVICE_ADDRESS;
             out STATUS     : UCL_RETURN);

procedure ROUTE_SWOP_TO_SAS
            (in ITEM    :  SW_COMMANDS_AND_RESPONSES;
```

```
             in SAS      : SAS_NAME;
             in OLD_SAS : SAS_NAME :=  NULL_SAS_NAME;
             out STATUS : UCL_RETURN);

procedure GET_CCSDS_PACKET_TYPE
            (in  SW_CMD      : SW_COMMAND;
             out PACKET_TYPE : CCSDS_PACKET_TYPE;
             out STATUS      : UCL_RETURN);

procedure SET_CCSDS_PACKET_TYPE
            (in  SW_CMD      : SW_COMMAND;
             in  PACKET_TYPE : CCSDS_PACKET_TYPE;
             out STATUS      : UCL_RETURN);


End GROUND_TO_OB_LIB;
```

I-2.3 Interface Description

These procedures control the emission of CCSDS command packets (Protocol Data Units or PDUs) from ground (e.g. AP) to onboard application (SWOP). The structure and the format of these packets are predefined in the database.

I-2.3.1 Test Node Initialization / Configuration Setup

I-2.3.1.1 SET_CCSDS_END_POINT

Specification

```
procedure SET_CCSDS_END_POINT
             (in GROUND_NODE: END_POINT;
              out STATUS: UCL_RETURN);
```

Description / Parameters

Sets up the CCSDS ground destination (End Point), which is simulated by the test node. From this end point, the addressing parameter for the CCSDS packets are derived when sending commands to onboard (e.g. sw–commands or FLAP execution requests) and receiving responses.

GROUND_NODE: pathname of the CCSDS End Point which shall be simulated by the test node.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Possible Return Codes:

OK	The procedure was successful
INVALID_NODE_NAME	GROUND_NODE refers to an unknown item
INVALID_TESTNODE_MODE	The test node is not in the right mode (NORMAL,SIMULATION)
RUNTIME_ERROR	Unexpected error

I-2.3.1.2 SET_DEVICE_ADDRESS

Specification

```
procedure SET_DEVICE_ADDRESS
           (in SOURCE      : END_POINT;
            in DESTINATION : END_POINT;
            in ADDRESS     : DEVICE_ADDRESS;
            out STATUS     : UCL_RETURN);
```

Description / Parameters

Set a new device address for the application id identified by the source node and the destination node. This device address will then be set in the physical address of the SWOP commands and response packets that are generated with that application id.

SOURCE : name of the source end point for identifying the application id.

DESTINATION : name of the destination end point for identifying the application id.

ADDRESS : string (up to 20 characters) to be set in the device_address field of the physical_address.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Possible Return Codes:

```
OK                       The procedure was successful
INVALID_NODE_NAME        There is no application id defined for
                         the end point pair (SOURCE, DESTINATION)
RUNTIME_ERROR            There was an error during the
                         interpretation of the UCL Intermediate Code
```

Example Call

```
SET_DEVICE_ADDRESS    (SOURCE: \apm\node,
                       DESTINATION: \ground\node,
                       ADDRESS: ”CC_BUS1”,
                       STATUS: stat);
```

I-2.3.1.3 ROUTE_SWOP_TO_SAS

Specification

```
procedure  ROUTE_SWOP_TO_SAS
               (in ITEM:   SW_COMMANDS_AND_RESPONSES;
               in SAS: SAS_NAME;
               in OLD_SAS: SAS_NAME
                               :=  NULL_SAS_NAME;
               out STATUS: UCL_RETURN)
```

Description / Parameters

Routes all items specified in ITEM which have the OLD_SAS_NAME assigned, to the new SAS specified by SAS_NAME.

ITEM : The item to be re–defined in the local memory of the test node. ITEM can be

– a single enditem of type SWOP_COMMAND or RESPONSE_PACKET

– an incomplete pathname pointing to a virtual node, thus redefining all SWOP_COMMAND or RESPONSE_PACKET in that subtree;

SAS: The short name (20 character) of the SAS (Specific Application Software) to be assigned to the ITEM. The SAS must run on the local test node .

OLD_SAS: The short name (20 character) of the SAS currently specified for item.
Only those items are taken into account for the route change, which match this SAS name. If OLD_SAS is equal to NULL_SAS_NAME, all enditems specified in ITEM are changed, regardless of their current SAS name.

STATUS : returns an integer, the UCL return status (see definition under `CONSTANTS`).

Example Call

ROUTE_SWOP_TO_SAS (\apm\cc_bus1\swops, ”CC_BUS_2_SAS”, ”CC_BUS_1_SAS”, stat);

I-2.3.2 Software Commanding to Onboard

I-2.3.2.1 ISSUE_SW_COMMAND

Specification

```
procedure ISSUE_SW_COMMAND
           (in  SW_CMD;SW_COMMAND ();
            in  ONBOARD_NODE: END_POINT;
            in  GROUND_NODE: END_POINT := \\;
            in  PRIO: PRIORITY := LOW;
            in  TIMEOUT: DURATION:= default_sw_command_timeout;
            in  MAX_DATA_LENGTH: DATA_LENGTH_RANGE := NO_DATA_LENGTH_LIMIT;
            out RECEIVED_DATA_LENGTH: DATA_LENGTH_RANGE;
            out TRANSACTION_ID: INTEGER;
            out RESULT: ONBOARD_RETURN;
            out STATUS: UCL_RETURN);
```

Description / Parameters

Builds a Telecommand – CCSDS Protocol Data Unit (PDU) – and initiates the uplink of the SWOP command. Onboard the command with the given input parameter will be executed.

NOTE: When setting the time–out to zero, only a GDU with the SW command in a CCSDS packet will be sent to SAS using a time–out value defined in the TES_CONFIG_FILE (under $TES_HOME/config) for the acknowledge of the GDU. The configurable value is defined under the name SW_ORDER.ISSUE_TIME-OUT_WHEN_NO_DELAY and has a predefined value of 5 sec (that can be configured by the user). No response will be acquired if the timeout is set to zero.

SW_CMD : name of the command to be executed onboard, followed by its list of actual parameters (if any)

ONBOARD_NODE : Onboard node (CCSDS_End_Point) where the command is sent to.

GROUND_NODE : Ground node which is simulated by CGS and where the response has to be returned to. If ”\\” is given, the value defined by SET_CCSDS_END_POINT is applicable.

PRIO (optional) : indicates the priority at which the SW–Command is to be executed within the Ground System.

TIMEOUT (optional) : the maximum time to wait for acknowledgement of this command (i.e. the time to wait until the onboard FLAP has finished)

MAX_DATA_LENGTH (optional) : the maximum length in octets of the packet data field (without checksum and headers). The data is truncated whenever the parameters imply a longer data field. as follows: For INTEGER the MSB of the integer and for STRING the maximum length will be written, string length is set accordingly to the truncated part. For other types, no data will be written.

RECEIVED_DATA_LENGTH : the length in octets of the out parameter part of the resonse packet (without checksum, headers, transaction id , onboard result). In case the received response packet is shorter than requested by the set of out parameters the remaining out parameters shall be set to their type respective null value (ie. INTEGER –> 0, REAL –> 0.0, TIME –>~:~, PATHNAME –> \\, STRING –> ””, STATECODE –> $OTHER). The STATUS will be set to RESPONSE_TOO_SHORT in this case. If a parameter cannot be filled completely then the parameter is also set to the null values as described above, except for INTEGER the remaining bytes are used as MSB of the integer and STRING which are truncated, ie. length is set to truncated string length.

TRANSACTION_ID: Returns the ID as received from the response packet: the primary header of the CCSDS uplink packet without the length field

RESULT: Returns the status of the execution as received from onboard (returned in response packet)

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Possible Return Codes:

```
OK                         The procedure was successful
BUSY                       The system (test node) is currently busy
                           with another SW command, try again later if necessary
INVALID_ITEM_NAME          The name of the SW Command is unknown
INVALID_NODE_NAME          The ID in ONBOARD_NODE or GROUND_NODE is unknown
INVALID_TESTNODE_MODE The test node is not in the right mode (NORMAL, SIMULATION)
ITEM_IS_DISABLED             The SW command is disabled
TIMEOUT                     There was no response from the SAS or
                            from the onboard system (response packet)
NO_GDU_SERVICE              The SAS did not announce the GDU Service
NO_ADU_SERVICE              The SAS did not announce the ADU Service
```

INVALID_APPLICATION_NAME	The SAS is not known
APPLICATION_NOT_READY	The SAS is not connected
APPLICATION_NACK	There was a negative response from the SAS
INVALID TIME	The timetag given in the GDU header or CCSDS header was not correct
PARAMETER_ERROR	When processing the parameter list of the SW command an error occurred
COMMUNICATION_ERROR	There was a problem in the communication to the SAS
RUNTIME_ERROR	There was an error during the interpretation of the UCL Intermediate Code
RESPONSE_TOO_SHORT	Response packet data length was shorter than expected.
MESSAGE_TOO_BIG	Parameter list too big.
OPERATION_ABORTED	Aborted (eg. through ABORT_AP, STOP_TES)
INVALID_ADU_TYPE	ADU of wrong type (e.g. not CCSDS packet) has been received as response
PROTOCOL_ERROR	Incorrect length of response packet or incorrect checksum in case when the checksum is checked, see configuration parameters TES.CHECK_CHECKSUM and TES.PROCESS_ON_INCORRECT_CHECKSUM

Example Call

ISSUE_SW_COMMAND (SW_CMD: \a\b\load_mon_tab (\x\mon_tab), ONBOARD_NODE: \apm\dms_1,
GROUND_NODE: \apm\cc, TRANSACTION_ID: t_id,
RESULT: onboard_status, STATUS:stat);

I-2.3.2.2 ENABLE_SW_COMMAND

Description / Parameters

Enables a SWOP Command for execution

Specification

```
procedure ENABLE_SW_COMMAND
            (in  SW_CMD: SW_COMMAND;
             out STATUS: UCL_RETURN);
```

SW_CMD : name of the SW–command to enabled.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Possible Return Codes:

OK	The procedure was successful This status is also returned if the item was already enabled
INVALID_ITEM_NAME	The name of the SW Command is unknown
INVALID_TESTNODE_MODE	The test node is not in the right mode (NORMAL,SIMULATION)
RUNTIME_ERROR	Unexpected error

Example Call

ENABLE_SW_COMMAND (\a\b\load_mon_tab, STATUS:stat);

I-2.3.2.3 DISABLE_SW_COMMAND

Description / Parameters

Disables a SWOP Command for execution (i.e. ISSUE command will be rejected for it)

Specification

```
procedure DISABLE_SW_COMMAND
            (in  SW_CMD: SW_COMMAND;
             out STATUS: UCL_RETURN);
```

SW_CMD : name of the SW–command to enabled.

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Possible Return Codes:

OK	The procedure was successful. This status is also returned if the item was already disabled
INVALID_ITEM_NAME	The name of the SW Command is unknown
INVALID_TESTNODE_MODE	The test node is not in the right mode (NORMAL,SIMULATION)
RUNTIME_ERROR	Unexpected error

Example Call

DISABLE_SW_COMMAND (\a\b\load_mon_tab, STATUS:stat);

I-2.3.3 FLAP Execution

I-2.3.3.1 EXECUTE_FLAP

Specification

```
procedure EXECUTE_FLAP
           (in  FLAP:FLAP_NAME ();
            in  ONBOARD_RECEPTION_NODE: END_POINT;
            in  ONBOARD_EXECUTION_NODE: END_POINT := \\;
            in  GROUND_NODE: END_POINT := \\;
            in  GROUND_PRIO: PRIORITY    := LOW;
            in  ONBOARD_PRIO: ONBOARD_PRIORITY
                               := default_onboard_priority;
            in  TIME_TAG: TIME    := ~:~;
            in  TIMEOUT:DURATION := default_flap_timeout;
            out TRANSACTION_ID: INTEGER;
            out RESULT: ONBOARD_RETURN;
            out ID: FLAP_ID;
            out FLAP_EXEC_STATUS: FLAP_RETURN;
            out STATUS: UCL_RETURN)
```

Description / Parameters

Builds a Telecommand – CCSDS Protocol Data Unit (PDU) – and initiates the uplink of the command. Onboard the command instantiates and starts a flight application program (FLAP) .
Waits for the response packet and returns the parameter (packet is returned after initiation of the FLAP)

NOTE: When setting the time–out to zero, only a GDU with the SW command in a CCSDS packet will be sent to SAS using a time–out value defined in the TES_CONFIG_FILE (under $TES_HOME/config) for the acknowledge of the GDU. The configurable value is defined under the name SW_ORDER.ISSUE_TIME-OUT_WHEN_NO_DELAY and has a predefined value of 5 sec (that can be configured by the user). No response will be acquired if the timeout is set to zero.

FLAP : name of the FLAP to be started, followed by its list of actual parameters (if any)

ONBOARD_RECEPTION_NODE : onboard node (CCSDS Endpoint) where the command is to be sent to

ONBOARD_EXECUTION_NODE:onboard node (CCSDS Endpoint) on which the FLAP shall be executed.

GROUND_NODE (optional) : Ground node which is simulated by CGS and where the response has to be returned to. If ”\\” is given, the value defined by SET_CCSDS_END_POINT is applicable.

GROUND_PRIO (optional) : indicates priority at which the command is to be sent in the Ground System

ONBOARD_PRIO(optional): indicates priority at which the FLAP is to be executed in the onboard system

TIME_TAG (optional) : indicates the time when to execute the FLAP.
The default value (~:~) indicates immediate execution (timetag is set to 0 in the uplink packet). The time value is processed by the onboard system

TIMEOUT (optional) : the maximum time to wait for acknowledgement of this command (i.e. the time until the response packet has been received).

ID : returns a unique identifier of this instance of the FLAP; 0 if the FLAP could not be started.

TRANSACTION_ID: Returns the ID as received from the response packet:
the primary header of the CCSDS uplink packet without the length field

RESULT: Returns the status of the execution as received from onboard (returned in response packet)

FLAP_EXEC_STATUS: Returns the status of the execution as received from onboard UCL Interpreter (returned in response packet)

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Possible Return Codes:

```
OK                          The procedure was successful
BUSY                        The system (test node) is currently busy
                            with another SW command, try again later if necessary
INVALID_ITEM_NAME           The name of the FLAP is unknown
INVALID_NODE_NAME           The ID in ONBOARD_NODE or GROUND_NODE is unknown
INVALID_TESTNODE_MODE       The test node isn't in right mode (NORMAL, SIMULATION)
ITEM_IS_DISABLED            The SW command defined for EXEC_FLAP is disabled
TIMEOUT                     There was no response from the SAS or
                            from the onboard system (response packet)
NO_GDU_SERVICE              The SAS did not announce the GDU Service
NO_ADU_SERVICE              The SAS did not announce the ADU Service
INVALID_APPLICATION_NAME    The SAS is not known
APPLICATION_NOT_READY       The SAS is not connected
APPLICATION_NACK            There was a negative response from the SAS
INVALID TIME                The timetag given in the GDU header or
                            CCSDS header was not correct
COMMUNICATION_ERROR         There was a problem in the communication to the SAS
SW_COMMAND_UNDEFINED        The SW Command, whose name/SID is defined
                            in the Configuration File, is not found in
                            the SW Command table, or it is even
                            not defined in the Configuration File
PARAMETER_ERROR             When processing the parameter list of the
                            FLAP command an error occurred
RUNTIME_ERROR               There was an error during the
                            interpretation of the UCL Intermediate Code
MESSAGE_TOO_BIG             Parameters to be put into CCSDS packet are too big
PROTOCOL_ERROR              Incorrect length of response packet or
                            incorrect checksum in case when the checksum
                            is checked, see configuration parameters
                            TES.CHECK_CHECKSUM and
                            TES.PROCESS_ON_INCORRECT_CHECKSUM
```

Example Call

EXECUTE_FLAP (FLAP:\ecls\fan1\start_fan (slow),
ONBOARD_RECEPTION_NODE: \apm\ node, ID: flap_id,
TRANSACTION_ID: t_id, RESULT: onboard_status,
FLAP_EXEC_STATUS: flap_stat, STATUS: stat);

I-2.3.3.2 EXECWAIT_FLAP

Specification

```
procedure  EXECWAIT_FLAP
            (in  FLAP: FLAP_NAME ();
             in  ONBOARD_RECEPTION_NODE: END_POINT;
             in  ONBOARD_EXECUTION_NODE: END_POINT := \\;
             in  GROUND_NODE: END_POINT := \\;
             in  GROUND_PRIO: PRIORITY    := LOW;
            in  ONBOARD_PRIO: ONBOARD_PRIORITY
                             := default_onboard_priority;
             in  TIME_TAG: TIME    := ~:~;
             in  TIMEOUT: DURATION := default_flap_timeout;
             out TRANSACTION_ID: INTEGER;
             out RESULT: ONBOARD_RETURN;
             out ID: FLAP_ID;
             out FLAP_EXEC_STATUS: FLAP_RETURN;
             out STATUS:UCL_RETURN)
```

Description / Parameters

Builds a Telecommand – a GDU containing a CCSDS Protocol Data Unit– and initiates the uplink of the command. Onboard the command instantiates and starts a flight application program (FLAP) . Waits for the response packet and returns the parameter (packet is returned after completion of the FLAP).

NOTE: When setting the time–out to zero, only a GDU with the SW command in a CCSDS packet will be sent to SAS using a time–out value defined in the TES_CONFIG_FILE (under $TES_HOME/config) for the acknowledge of the GDU. The configurable value is defined under the name SW_ORDER.ISSUE_TIME-OUT_WHEN_NO_DELAY and has a predefined value of 5 sec (that can be configured by the user). No response will be acquired if the timeout is set to zero.

FLAP : name of the FLAP to be started, optionally followed by its list of actual parameters.

ONBOARD_RECEPTION_NODE : onboard node (CCSDS Endpoint) where the command is to be sent to

ONBOARD_EXECUTION_NODE: onboard node (CCSDS Endpoint) on which the FLAP shall be executed.

GROUND_NODE (optional) : Ground node which is simulated by CGS and where the response has to be returned to. If ”\\” is given, the value defined by SET_CCSDS_END_POINT is applicable.

GROUND_PRIO (optional) : indicates priority at which the command is to be sent in the Ground System

ONBOARD_PRIO (optional) : indicates priority at which the FLAP is to be executed in the onboard system

TIME_TAG (optional) : indicates the time when to execute the FLAP. The default value (~:~) indicates immediate execution (timetag is set to 0 in the uplink packet). The time value is processed by the onboard system

TIMEOUT (optional) : the maximum time to wait for acknowledgement of this command (i.e. the time to wait until the response packet has been received)

ID : returns a unique identifier of this instance of the FLAP; 0 if the FLAP could not be started.

TRANSACTION_ID: Returns the ID as received from the response packet: the primary header of the CCSDS uplink packet without the length field

RESULT: Returns the status of the execution as received from onboard (returned in response packet)

FLAP_EXEC_STATUS:
Returns the status of the execution as received from onboard UCL Interpreter (returned in response packet)

STATUS : returns an integer, the UCL return status (see definition under CONSTANTS).

Possible Return Codes:

```
OK                          The procedure was successful
BUSY                        The system (test node) is currently busy
                            with another SW command, try again later if necessary
INVALID_ITEM_NAME           The name of the FLAP is unknown
INVALID_NODE_NAME           The ID in ONBOARD_NODE or GROUND_NODE is unknown
INVALID_TESTNODE_MODE       The test node is not in the right mode
                            (NORMAL, SIMULATION)
ITEM_IS_DISABLED            The SW command defined for EXECWAIT_FLAP  is disabled
TIMEOUT                     There was no response from the SAS or
                            from the onboard system (response packet)
NO_GDU_SERVICE              The SAS did not announce the GDU Service
NO_ADU_SERVICE              The SAS did not announce the ADU Service
INVALID_APPLICATION_NAME    The SAS is not known
APPLICATION_NOT_READY       The SAS is not connected
APPLICATION_NACK            There was a negative response from  the SAS
INVALID TIME                The timetag given in the GDU header or
                            CCSDS header was not correct
COMMUNICATION_ERROR         There was a problem in the communication to the SAS
SW_COMMAND_UNDEFINED        The SW Command, whose name/SID is defined
                            in the Configuration File, is not found in
                            the SW Command table, or it is even
                            not defined in the Configuration File
PARAMETER_ERROR             When processing the parameter list of the
                            FLAP command an error occurred
RUNTIME_ERROR               There was an error during the
                            interpretation of the UCL Intermediate Code
MESSAGE_TOO_BIG             Parameters to be put into CCSDS packet are too big
PROTOCOL_ERROR              Incorrect length of response packet or
                            incorrect checksum in case when the checksum
                            is checked, see configuration parameters
                            TES.CHECK_CHECKSUM and
                            TES.PROCESS_ON_INCORRECT_CHECKSUM
```

Example Call

EXECWAIT_FLAP (FLAP:\ecls\fan1\start_fan (slow),
ONBOARD_RECEPTION_NODE:\apm\ node, GROUND_PRIO:low,
TRANSACTION_ID: t_id, RESULT: onboard_status, ID:flap_id,
FLAP_EXEC_STATUS: flap_return, STATUS:stat);

J CGS SYSTEM LIMITATIONS

J-1 System Table Sizes

The following limitations exist for the size of system tables:

Number of enditems in ADU (CCSDS and Unstructured):	750
Number of enditems in ADU (Structured)	100 *)

NOTE: *) The number of enditems in a structured ADU is also limited by the size of the ADU (see below). In case of EGSE_BYTE_STREAM_MEASURMENTS (size = 256 characters effectively) not more than 15 enditems can be defined in a structured ADU otherwise the maximum size as defined below would be violated.
Furthermore the selection to have a physical address defined for each measurement reduces the maximum number of enditems to 68 only (between 68 and 100, dependent on the physicall address information given)

Number of online parameters for telecommands / binary packets	255
Number of predefined parameters for Predefined TC and Binary Packets	255
Number of online parameters for software commands	255
Size of a CCSDS Packet (bytes)	4096
Number of enditems in a synoptic picture (configurable in hci.ini)	50
Size of the data part of an ADU (bytes)	4096
Number of statecodes per discrete calibration	256
Length of a calibrated String (bytes)	255
Number of limit sets per enditem	5
Number of entries in the system topology table	80
Number of enditems in a monitor list	500
Number of enditems in a GDU list	500
Number of entries in a user message description table (some 200 are used for CGS itself from a predefined file)	1500
Number of raw values in the simulated value table	1000
Number of ADUs of type UNSTRUCTURED or CCSDS in the simulated value table	100

Number of CDUs loadable into each test node	500
Maximum Size of Source Code for Automated Procedures (bytes)	300 K
Maximum Size of I– Code at compile time (bytes)	TBD (>32) K
Maximum Size of I– Code at runtime for Automated Procedures (bytes) (configurable: COMMON_POOL.STACK_SIZE in TES_CONFIG_FILE)	400 K
Max Number of Variables in Global Data Area of User Libraries and APs per compilation unit (depending on the variable types).	128..256
Number of active APs on a test node (configurable in TES_CONFIG_FILE)	40
Number of SAS on a test node	20

J-2 Resource Dependent Limitations

<u>Number of enditems loadable into each test node</u>

The number is virtually unlimited, but depends on memory available for the TES process.

On HP, the MAXDSIZE parameter of the HP_UX operating system might need update. Also some configuration parameters in the TES_CONFIG_FILE might need adaptation.

In CGS_4.2.0.11 it has been **verified, that 26000** enditems can be loaded having the following parameter set:

HP:
- 256 MB RAM
  The TES process uses at least 150 MB in this configuration
- HP_UX Kernel Parameter MAXDSIZ is set to 200 MB
- In TES_CONFIG_FILE:
  - TES_RPI.TIME_OUT_INIT 1_500_000
    (> 20 min loading time needed)
  - DATA_PROCESSOR.STACK_SIZE 1_200_000

SUN
- 256 MB RAM
  The TES process uses at least 165 MB in this configuration
- In TES_CONFIG_FILE:
  - TES_RPI.TIME_OUT_INIT 1_500_000
    (> 15 min loading time needed)
  - DATA_PROCESSOR.STACK_SIZE 1_200_000
  - TES_CON_EXEC.STACK_SIZE 1_200_000

J-3 Miscellaneous Resources

During run time, Server disk space storage can be affected, eg by temporary files. Such temporary files will be created in specific file system temporary areas (eg /tmp). The temporary files will be created during standard process creation and during various software tasks eg file editing, compilation, temporary log files etc. The temporary areas should also allow for the dumping of system (UNIX) software during system errors. To cover for these eventualities, 100 MB to 200 MB should be allocated to the appropriate file systems to allow for temporary files in /tmp.

K USER DEFINABLE CONFIGURATION PARAMETER FOR CGS

CGS Configuration Parameter definable by the user

The configurations files are located under the following location :
$HCI_HOME/config for hci.ini
$TES_HOME/config for TES_CONFIG_FILE
$TEV_HOME/config for tev_configuration_file.dat
$DBS_HOME/config for dbs_configuration_file.def
$DBS_HOME/config for fa_sas_configuration_file.def

General Description of $HCI_HOME/config/hci.ini

Version: @(#) hci.ini /main/cgs_4.1.1patches/cgs_4.2/cgs_4.3/3 03/27/00 16:25:43@(#)

The Online Test Control initialization file hci.ini is used to configure the Online Test Control software at initialization time. When started the file hci.ini located at $HCI_HOME/config is read and Online Test Control is initialized according to the definitions in hci.ini.

The hci.ini file is separated into groups which may have attributes. A group is indicated by a name enclosed by square brackets, an attribute definition consists of the attributes name followed by the assignment operator = and the attributes value. An attribute definition is terminated by the end of line, i.e. it is not allowed to continue a definition on the next line. Comments may be inserted everywhere in the hci.ini file, but only on separate lines.
A comment line must always start with a sharp # and is terminated by the end of line.

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
	General Test Execution			
TES.ENABLE_ACQUISITION_TIMEOUT	Timeout for ack from SAS upon disable ADU request, seconds	TES_CONFIG_FILE	60.0	0.0 .. 86400.0
TES.DISABLE_ACQUISITION_TIMEOUT	Timeout for ack from SAS upon disable ADU request, seconds	TES_CONFIG_FILE	60.0	0.0 .. 86400.0
TES.LOAD_APPLICATION_TIMEOUT	Timeout for connect from SAS after being started, seconds	TES_CONFIG_FILE	30.0	0.0 .. 86400.0
TES.INIT_APPLICATION_TIMEOUT	Timeout for ack from SAS upon init application, milliseconds	TES_CONFIG_FILE	60000	0 .. 86400000
TES.START_APPLICATION_TIMEOUT	Timeout for ack from SAS upon start application, milliseconds	TES_CONFIG_FILE	60000	0 .. 86400000
TES.RESET_APPLICATION_TIMEOUT	Timeout for ack from SAS upon reset application, milliseconds	TES_CONFIG_FILE	60000	0 .. 86400000
TES.GET_APPLICATION_STATUS_TIMEOUT	Timeout for ack from SAS upon get application status, milliseconds	TES_CONFIG_FILE	60000	0 .. 86400000
TES.WRITE_MESSAGE_TO_APPLICATION_TIMEOUT	Timeout for ack from SAS upon write message to application, milliseconds	TES_CONFIG_FILE	60000	0 .. 86400000
TES.DOWNLOAD_FILE_TO_APPLICATION_TIMEOUT	Timeout for ack from SAS upon download file to application, milliseconds	TES_CONFIG_FILE	60000	0 .. 86400000
TES.INITIALISE_TIMEOUT	Timeout for ack from another TES in initialisation exchange of distribution tables, milliseconds	TES_CONFIG_FILE	30000	0 .. 86400000
TES.DEFAULT_WORKSTATION	Default workstation (HCI) for TES output in emergency case. Must be a logical name for a workstation (see SYSTEM_TOPOLOGY_TABLE).	TES_CONFIG_FILE	HCI_01	Logical Name (1..20 char)

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
API_CONTROLLER.TIMEOUT_PERIOD_FOR_READ_CMD	If an SAS does not call read_command within this time period [seconds], then TES will automatically disconnect the SAS	TES_CONFIG_FILE	30.0	1.0 .. 86400.0
TES_RPI.TIME_OUT_INIT	Timeout value [milliseconds] for the TES INIT operation of TES, i.e. the time TSCV will wait until TES has been initialised. Should be changed, when loading very large test configurations	TES_CONFIG_FILE	300000	0 .. maxint
DISTRIBUTION_TABLE.MAX_NB_SID	Maximal number of enditems in the test configuration. This number constrains a list that is used for remote operations on measurements / sw–variables, GDU's and GDU lists.	TES_CONFIG_FILE	20000	1000 .. 100000
REQUEST_FETCHER.NO_OF_MESSAGE_AGENTS	The number of message_agent tasks, i.e. the max number of requests from TES_RPI or TES_API being served in TES_Core in parallel	TES_CONFIG_FILE	10	1 .. 32
	UCL Execution			
UCLI_CONTROLLER.NUMBER_OF_UCL_INTERPRETER	number of ap interpreters. This determines the number of AP that can run in parallel	TES_CONFIG_FILE	20	1 .. 40
UCLI_CONTROLLER.MAXIMUM_NUMBER_OF_EMERGENCY_APS	The number of slots that shall be reserved for emergency APs	TES_CONFIG_FILE	5	1 .. number of UCL interpreter
UCLI_CONTROLLER.HK_VALUE_UPDATE_PERIOD	Period for writing the "current UCL statement" HK value, seconds	TES_CONFIG_FILE	5.0	1.0 .. 86400.0
COMMON_POOL.STACK_SIZE	Defines, among others, the size needed to execute UCL I–Code for APs. Increase for executing very large AP's.	TES_CONFIG_FILE	500000	>= 500000
AP_CONTROLLER.STACK_MACHINE_INITIATOR.STACK_SIZE	Defines, among others, the size needed to execute UCL I–Code for APs. Increase for loading very large AP's.	TES_CONFIG_FILE	40000	>= 40000
STACK_MACHINE.DEBUG	Enables debug output from I–code interpretation; one file per AP, in directory $TES_HOME/data with prefix "ap"	TES_CONFIG_FILE	false	true, false
	Data Processing			
ADU_QUEUE.MAX_NUMBER_OF_QUEUED_ADUS	Max number of ADUs that can be queued for processing before TES starts throwing away	TES_CONFIG_FILE	50	1.. 100
DATA_PROCESSOR.MEASUREMENT_TIME_STAMP_IN_LT	Indicates if the time stamp of the measurements (accessible via UCL) is based on local time (true) or SMT (false).	TES_CONFIG_FILE	true	true, false
TES.CHECK_CHECKSUM	allow or inhibit the checking of the checksum in ADU of type CCSDS packet or in response packet for software commanding. If it is set to true and the checksum is not detected to be correct, an error message is generated	TES_CONFIG_FILE	false	true, false
TES.PROCESS_ON_INCORRECT_CHECKSUM	allow or inhibit the processing of measurements contained in ADU of type CCSDS packets or the processing of response packets for software commanding if the checksum is incorrect. This parameter is only used in the case where TES.CHECK_CHECKSUM is set to true	TES_CONFIG_FILE	true	true, false

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
MONITOR.ACTION_HANDLERS	Number of action_handlers in the monitor, i.e. the number of monitor actions (GDUs or APs) that can be handled in parallel	TES_CON-FIG_FILE	5	1 .. 20
ACTION_BUFFER.ACTION_BUFFER_BUFFER_LENGTH	The length of the action_buffer data structure	TES_CON-FIG_FILE	100	100 .. 1000
DATA_PROCESSOR.MESSAGE_ON_CONDITION	allow or inhibit the generation of messages to the event log for each triggered condition	TES_CON-FIG_FILE	true	true, false
DATA_PROCESSOR.ENABLE_ALL_ADUS_IN_REPLAY	Enables all ADUs for replay in the init–call (i.e. when setting TES to REPLAY mode)	TES_CON-FIG_FILE	false	true, false
ADU_GENERATOR_POOL.MAX_NO_OF_AGENTS	The max number of adu_generator tasks, i.e. the max number of different adus that can be simulated in parallel	TES_CON-FIG_FILE	100	10 .. 200
	<u>Housekeeping Values</u>			
HK_VALUE_PROVIDER.UPDATE_CLOCK_PERIOD	Time period [seconds] for cyclical update of the HK value for LT	TES_CON-FIG_FILE	60.0	1.0 .. 86400.0
HK_VALUE_PROVIDER.UPDATE_DISK_PERIOD	Time period [seconds] for cyclical update of the HK value for free disk space	TES_CON-FIG_FILE	30.0	1.0 .. 86400.0
HK_VALUE_PROVIDER.UPDATE_DBS_PERIOD	Time period [seconds] for cyclical update of the DBS (TRDB) related HK values	TES_CON-FIG_FILE	60.0	1.0 .. 86400.0
HK_VALUE_PROVIDER.UPDATE_TSS_PERIOD	Time period [seconds] for cyclical update of the TSS (time) related HK values	TES_CON-FIG_FILE	60.0	1.0 .. 86400.0
	<u>GDU Handling</u>			
GDU_TABLE.TAI_EPOCH_START_YEAR	The CCSDS–Recommended epoch start time (TAI): year	TES_CON-FIG_FILE	1980	0 .. current year
GDU_TABLE.TAI_EPOCH_START_MONTH	The CCSDS–Recommended epoch start time (TAI): month	TES_CON-FIG_FILE	1	1..12
GDU_TABLE.TAI_EPOCH_START_DAY	The CCSDS–Recommended epoch start time (TAI): day	TES_CON-FIG_FILE	6	1..31 (val-id day)
GDU_HANDLER.ISSUE_TIMEOUT_WHEN_MONITORING_EXCEPTION	Default timeout value when issuing gdus upon a monitoring exception, seconds	TES_CON-FIG_FILE	5.0	0.05 .. 100.0
GDU_HANDLER.TC_VERIFI-CATION_DISABLED	Default value for disable TC Verification: Sup-press Verification for all TC sent independent of definitions stored in MDB	TES_CON-FIG_FILE	false	false,true
REQUEST_CONNEC-TOR.USE_TYPE_IN_CCSDS_SEQ_COUNT	For CCSDS packets, use the type field of the pri-mary header to determine the sequence counter in a command (i.e. maintain sequence counter per applicaion_id and type, if true)	TES_CON-FIG_FILE	true	false, true
	<u>Replay Mode</u>			
REPLAYER.GDU_MESSAGES	Allows to suppress generation of messages per GDU when replaying data: FALSE: no messages is generated	TES_CON-FIG_FILE	TRUE	TRUE, FALSE
SMT_UPDATE_PERIOD_IN_REPLAY	The frequency to update the SMT clock in replay mode, in second	TES_CON-FIG_FILE	5.0	1.0 .. 60.0

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
SW_CMDER.EXEC_FLAP_SWOP_PATHNAME	Pathname for the SW command to be used for starting a FLAP in the onboard system without waiting for the FLAP to terminate.	TES_CONFIG_FILE	no valid default	Pathname to SW Cmd in MDB
SW_CMDER.EXECWAIT_FLAP_SWOP_PATHNAME	Pathname for the SW command to be used for starting a FLAP in the onboard system with subsequent waiting for the FLAP to terminate	TES_CONFIG_FILE	no valid default	Pathname to SW Cmd in MDB
SW_CMDER.CHECK_TRANSACTION_ID	Check that the transaction id of the response packet matches the CCSDS primary header of the SW command sent	TES_CONFIG_FILE	false	false, true
SW_CMDER.ISSUE_TIMEOUT_WHEN_NO_DELAY	Default timeout for telecommand part value when issuing a software command with null delay, seconds	TES_CONFIG_FILE	5.0	0.05 .. 100.0
ARCHIVE.LIMIT_DISK_SPACE	Limit for issuing warning concerning the amount of free disk space (in bytes)	TES_CONFIG_FILE	20000	0..maxint
REPLAY_TIME_FOR_EVENT_TIME_STAMPS	Indicates if the log events / messages produced during a replay session use the replay local time or the current local time.	TES_CONFIG_FILE	false	false,true
	<u>Test Evaluation</u>			
GENERATION	location of the data set generation INTERNAL – data set generated by TEV. EXTERNAL – external generation (TDCS,...)	tev_configuration.data	INTERNAL	INTERNAL, EXTERNAL
DATABASE	raw data source used for internal data set generation TRDB – raw data provided by TRDB. FOREIGN – raw data provided by other database (TDCS,...)	tev_configuration.data	TRDB	TRDB, FOREIGN
LIMIT_ARCHIVE_FILES_SIZE	Define a limit size for the archive files. Only used to display a warning message before performing a raw data dump, a data set or before searching the list of ADUs and GDUs dumped in archive files. The size is in Megabytes	tev_configuration.data	30 Mbytes	*positiv integer*
EXCEL_SEPARATOR	Define the character to be used as separator for the excel files.	tev_configuration.data	,	special character
	<u>Test Setup and Configuration</u>			
DEBUG_TRACING_IS_ON	Enables debug tracing for TSCV program	tscv_configuration_file.dat	FALSE	TRUE/ FALSE
DEBUG_TRACE_FILE_NAME	name for logfile of debug tracing for TSCV program	tscv_configuration_file.dat	tscv_debug_tscv.trace	any file name
DEBUG_TRACE_ONLY_TO_FILE	trace to logfile only for TSCV program	tscv_configuration_file.dat	TRUE	TRUE/ FALSE

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
	Test Result Database			
PERCENT_OF_FREE_SPACE	PERCENT_OF_FREE_SPACE is used to define TRDB status DISK_FULL. If disk free space <= PERCENT_OF_FREE_SPACE in the disk partition defined by VICOS_CEN_DBS_HOME then TRDB disk status is DISK_FULL	dbs_configuration_file.def	2	1..100
CRITICAL_SPACE SECURE_SPACE	Parameters for Automatic Archiving : The parameters CRITICAL_SPACE and SECURE_SPACE are used to define the level at which automatic archiving will start and stop. If CRITICAL_SPACE = 4 and SECURE_SPACE = 6 then automatic archiving will start when free disk space is 4% of the total disk space and automatic archiving will stop when free disk space is 6% of the total disk space. Note SECURE_SPACE must always be greater than CRITICAL_SPACE Parameter CRITICAL_SPACE is also used by the following DBS operations: retrieve_execution_session_data retrieve_evaluation_session import_execution_session import_evaluation_session The operations will fail with status TABLE_SPACE_FULL or FILE_SYSTEM_FULL if CRITICAL_SPACE is exceeded as a result of the operation. Please note that the values have been increased by 2, both, in order not to endanger file transmission from the local nodes to the TRDB disk while the FA SAS copies files from TRDB to the final archive data medium.	dbs_configuration_file.def	4 / 6	1..100
EVENT_TABLE_OWNER	used to prefix the event table names in sql statements. For example 'select * from OPS$cgsadmin.EVENT_EXEC_0024'; These tables are created by the central dbs processes. This means that the owner of these tables is the same as the user running central dbs processes.The following constant should be defined with the following syntax : "OPS$unix_user_name." where 'unix_user_name' is the name of the user running central dbs processes (CAUTION : the "." is required, 'OPS' in upper case). During "install_dbs" "cgsadmin" will be substituted by the onsite DBS_OWNER UNIX username. See $DBS_HOME/util/common/install_dbs. Name must be identical to DBS_OWNER_NAME in fa_sas_configuration_file.def	dbs_configuration_file.def	OPS$cgsadmin.	OPS$<UNIX user name>.

Parameter Name	***Meaning***	***config file***	***Default***	***Possible values***
ONL_EVAL_EVT_PERIOD ARCHIVE_EVENT_EVL_PERIOD	Parameter to control rate of files generated by DBS for event logging: ARCHIVE_EVENT_PERIOD is the delay between cyclic lookups into the event buffer within the DBS local software. The items in the buffer are written to the file at each lookup. In case ONL_EVAL_EVT_PERIOD is expired at a lookup, or if the file to be generated contains more than MAX_EVT_NUMBER_IN_LOCAL_FILE items, the file is closed and sent to Central DBS for further processing, and a new file is opened. ONL_EVAL_EVT_PERIOD determines the time a user has to wait until event logs are available for (online) evaluation via the TEV Tool. Attention:Parameter DELAY_CLOSED_LOOP must be set accordingly ! For systems with high rates of event logging, it should be noted, that DBS internally keeps the logitems in memory buffers before they are written to the files. On testnodes, there are TN_EVT_BUFFER_SIZE items storable in these buffers. A special buffersize for SUN applications (HCI, TSCV,...) has been introduced in order to avoid unused large buffers. A number of buffers is maintained in the latter case to store events, the max. number of events to store is: LOCAL_EVENT_POOL_SIZE * EVENT_BUFFER_SIZE To avoid overflow of these buffers (”POOL SATURATED”), the parameters above must be set accordingly, meaning that the buffer lookup delay (ARCHIVE_EVENT_EVL_PERIOD) must be decreased, if the logging rate is higher than the storable items/second.	dbs_configuration_file.def	10 s 100 ms	1 ... maxint 10 ... maxint

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
ONL_EVAL_EVL_PERIOD ARCHIVE_EVENT_EVL_PERIOD	Parameter to control rate of files generated by DBS for engineering value logging: ARCHIVE_EVL_PERIOD is the delay between cyclic lookups into the evl buffer within the DBS local software. The items in the buffer are written to the file at each lookup. In case ONL_EVAL_EVL_PERIOD is expired at a lookup, or if the file to be generated contains more than MAX_EVL_NUMBER_IN_LOCAL_FILE items, the file is closed and sent to Central DBS for further processing, and a new file is opened. ONL_EVAL_EVL_PERIOD determines the time a user has to wait until engineering value logs are available for (online) evaluation via the TEV Tool. Attention:Parameter DELAY_CLOSED_LOOP must be set accordingly ! For systems with high rates of engineering value logging, it should be noted, that DBS internally keeps the logitems in memory buffers before they are written to files. There are EVL_BUFFER_SIZE items storable in these buffers.To avoid overflow of these buffers (”POOL SATURATED”), the parameters above must be set accordingly, meaning that the buffer lookup delay (ARCHIVE_EVENT_EVL_PERIOD) must be decreased if the logging rate is higher than EVL_BUFFER_SIZE items/second.	dbs_configuration_file.def	10 s 100 ms	1 ... maxint 10 ... maxint
DELAY_CLOSE_LOOP	Parameter to control the time DBS is waiting in a Close_Test_Session operation, until all clients have sent their data to Central_DBS.	dbs_configuration_file.def	3000 ms	1000 ... maxint
MAX_EVL_FILE_SIZE	The maximum size (in bytes) of files where engineering values are strored within the TRDB. Influences the number of files generated within the TRDB.	dbs_configuration_file.def	1 MB	255 ... maxint
LOCAL_EVENT_POOL_SIZE EVENT_BUFFER_SIZE	These parameters are valid for SUN applications (HCI, TSCV,...) only. LOCAL_EVENT_POOL_SIZE defines the number of buffers within the pool for event logging. EVENT_BUFFER_SIZE defines the number of event items in a buffer. Thus LOCAL_EVENT_POOL_SIZE * EVENT_BUFFER_SIZE items can be stored in the buffers, before they are written to files. In case the buffers are overflowing, the items are discarded and a message is generated to the user (”POOL SATURATED”). A special buffersize for SUN applications has been introduced in order to avoid unused large buffers (see description of TN_EVT_BUFFER_SIZE).	dbs_configuration_file.def	4 25	1 ... maxint 1 ... maxint

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
TN_EVT_BUFFER_SIZE	This parameter is valid for testnode applications (TES) only. TN_EVT_BUFFER_SIZE defines the number of events which can be stored, before they are written to files. In case the buffer is overflowing, the items are discarded and a message is generated to the user (”POOL SATURATED”).	dbs_configuration_file.def	2000	1 ... maxint
MAX_EVT_NUMBER_IN_LOCAL_FILE	This value denotes the maximum number of events to be stored in a local file. The file is automatically closed and sent to the central DBS if this value is exceeded, or if ARCHIVE_EVENT_EVL_PERIOD expires.	dbs_configuration_file.def	1000	1 ... maxint
EVL_BUFFER_SIZE	EVL_BUFFER_SIZE defines the number of engineering values which can be stored, before they are written to files. In case the buffer is overflowing, the items are discarded and a message is generated to the user (”POOL SATURATED”)	dbs_configuration_file.def	2000	1 ... maxint 1 ... maxint
MAX_EVL_NUMBER_IN_LOCAL_FILE	This value denotes the maximum number of engineering values to be stored in a local file. The file is automatically closed and sent to the central DBS if this value is exceeded, or if ARCHIVE_EVENT_EVL_PERIOD expires.	dbs_configuration_file.def	2000	1 ... maxint
EVENT_FILENAME_BUFFER_SIZE	This value denotes the number of filenames to be stored within the shared memory region between the central DBS exec process and the DBS event handler. The name of the file containing the events produced by the local nodes is transmitted to the event handler where the file is evaluated and its contents is stored in the database. A value of 200 for a CGS configuration with 5 testnodes each producing up to 100 events per second should be sufficient. Please note that no files will be lost in case the limit of this buffer is reached. The remaining filenames are stored within the central exec process then!	dbs_configuration_file.def	200	1 ... maxint
EVL_FILENAME_BUFFER_SIZE	This value denotes the number of filenames to be stored within the shared memory region between the central DBS exec process and the DBS file handler. The name of the file containing either engineering value logs or raw data produced by the local nodes is transmitted to the file handler where the file is evaluated and its contents is stored on the central disk. A value of 100 for a CGS configuration with 5 testnodes each producing up to 100 EVL per second and 2 raw data files of size 50 MB per hour should be sufficient. Please note that no files will be lost in case the limit of this buffer is reached. The remaining filenames are stored within the central exec process then!	dbs_configuration_file.def	100	1 ... maxint

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
AUTO_AR-CHIVE_EVENT_TABLE_SIZE	In case the size of an Event Table exceeds this limit it is flushed into a file. This file is then requested to be stored inside TRDB (part of auto archiving). The value is expressed in bytes. Please note that the size of the stored eventfile may exceed this value, because the size of the ORACLE Event Table is checked cyclically in equidistant periods.	dbs_configura-tion_file.def	10 MB	1 ... maxint
ORA_EVENT_BUFFER_SIZE	Number of Events in an Oracle buffer to accelerate the Event insertion. The events are inserted by buffers, i.e. a complete buffer is stored within the database rather than the single events.	dbs_configura-tion_file.def	300	1 ... maxint
COMMS_CENTRAL_TIME_OUT	Central DBS processes (Central Exec, Central Eval and Central Arch) access the Communication Services message buffers. Time out associated with reading these buffers is given by this value. If COMMS_CENTRAL_TIME_OUT is exceeded then the read will be aborted. No error message or status is associated with this. Message buffers will be read again as per the polling time. Please note: the timeout value has been set to 2 seconds, i.e. 2000 ms, in order to enable the central processes to catch a SIGTERM signal (on command ”kill –TERM <pid>”). This signal is given on ”CGS shutdown” and is used to ensure a proper shutdown with removal of shared memory regions.	dbs_configura-tion_file.def	2000 ms	1 ... maxint
FA_SAS_REPLY_TIME_OUT	The time out to wait for an ACK from FA–SAS on the Central Archive side (expressed in milliseconds) ATTENTION: This time is connected with the FA_SAS_MMI_TIME_OUT time defined in the configuration file of the FA–SAS (i.e. FA_SAS_REPLY_TIME_OUT must be bigger than the FA_SAS_MMI_TIME_OUT).	dbs_configura-tion_file.def	240000	1..max-int
NUMBER_OF_FA_DRIVE_AVAILABLE	The number of drive available for Final Archive	dbs_configura-tion_file.def	2	1 or 2
FA_DEVICE_FILE_NAME 1	The filename applicable for the device 1. The device file names will be used as link to the ”real” MO–device (example: /dev/mo1 is a symbolic link to /dev/dsk/c0t3d0s2)	fa_sas_configu-ration_file.def	/dev/mo1	UNIX filename
FA_DEVICE_FILE_NAME 2	The filename applicable for the device 2. The device file names will be used as link to the ”real” MO–device (example: /dev/mo2 is a symbolic link to /dev/dsk/c0t4d0s2)	fa_sas_configu-ration_file.def	/dev/mo2	UNIX filename
FA_DEVICE_ROOT	The root part of the device filenames (root name must be identical to the name used in device name)	fa_sas_configu-ration_file.def	/dev	UNIX filename

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
FA_MMI_TIMEOUT	The time the FA_SAS is waiting for a user to answer an exchange request	fa_sas_configuration_file.def	120	1..max-int
DBS_OWNER_NAME	The Name of the DBS Owner and privileged user: The files on the TRDB disk and on the FA disk are owned by DBS Owner. During "install_dbs" the ownername "DBS OWNER NAME DEFAULT" will be substituted by the onsite DBS OWNER UNIX username. See $DBS_HOME/util/common/install_dbs. DBS_OWNER_NAME must be identical to 'DBS OWNER NAME DEFAULT' in dbs_configuration_file.def	fa_sas_configuration_file.def	cgsadmin	UNIX User-name
	<u>Online Test Control (HCI)</u>			
[ONLINE_TEST_CONTROL]CommunicationRequired	Defines whether Online Test Control requires OK status when setting up the communication software. If True, it will abort execution on setup errors.	hci.ini	True	Boolean (FALSE, DISABLE, NO, OFF,TRUE, ENABLE, YES, ON)
[ONLINE_TEST_CONTROL]DataRequestRetryDelay	If data can't be requested (e.g. for the AP Status) Online Test Control will retry the request after a.m. delay.	hci.ini	60 seconds	Dureation (0..86400)
[ONLINE_TEST_CONTROL]DataRegistrationTimeOut	Defines the time in seconds HCI will wait for the test node until it registers its data (data distribution table)	hci.ini	60 seconds	Duration
[ONLINE_TEST_CONTROL]EndItemTimeOut	Minimum number of seconds Online Test Control will wait until an end item value is delivered from a test node (HLCL commanding).	hci.ini	20.0	Duration
[ONLINE_TEST_CONTROL]ExitTimeOut	Time out after which Online Test Control will ask for a forced shut down.	hci.ini	45 sec.	Duration
[ONLINE_TEST_CONTROL]MaxAPInputDialogs	Maximum number of AP Input Dialogs that can be displayed in parallel.	hci.ini	2	Positive (1..tdb)

Parameter Name	***Meaning***	***config file***	***Default***	***Possible values***
[ONLINE_TEST_CONTROL]MaxAPCommands	Maximum number of AP commands (e.g. start/abort AP, execute library routine) that can be executed in parallel.	hci.ini	10	Positive
[ONLINE_TEST_CONTROL]MaxDeliveryNotes	Maximum number of delivery notes (data requests and cancel notes) that can be handled.	hci.ini	500	500..tdb
ONLINE_TEST_CONTROL MaxEndItemRequests	Maximum number of enditems that can be queried by HLCL commands at the same time.	hci.ini	10	Positive
[ONLINE_TEST_CONTROL]MaxTestNodes	Maximum number of test nodes supported by Online Test Control	hci.ini	4	1..32
[ONLINE_TEST_CONTROL]MaxWindowsPerApplication	Maximum number of window instantiations of each kind of application (e.g. command facility, AP status display). A MaxWindow attribute (see window application groups, e.g. MaxWindow of [AP_STATUS] must never exceed this constant.	hci.ini	32	1..32
[ONLINE_TEST_CONTROL]NonInterruptableDBAccess	Defines the method of synchronization of mission database access.	hci.ini	False	Boolean
[ONLINE_TEST_CONTROL]APClientTimeOut	Internal configuration item.	hci.ini	Do not modify/ set.	
[ONLINE_TEST_CONTROL]HomeDirectory	Defines the home directory.	hci.ini	Do not modify/ set.	
[ONLINE_TEST_CONTROL]ConfigDirectory	Defines the data directory.	hci.ini	Do not modify/ set.	
[ONLINE_TEST_CONTROL]DataDirectory	defines the data directory	hci.ini	Do not modify/ set.	
[ONLINE_TEST_CONTROL]IconDirectory	Defines the icon directory. Can be used to define own icons for Online Test Control applications.	hci.ini	$HCI_HOME/config	Text (directory path-name)
[ONLINE_TEST_CONTROL]DataViewsSearchPath	Defines the DataViews search path.	hci.ini	Do not modify/ set.	
[ONLINE_TEST_CONTROL]MPS_Directory	Defines the MPS home directory.	hci.ini	Do not modify/ set.	
[ONLINE_TEST_CONTROL]VersionIdTable	Defines the location of the version id table.	hci.ini	Do not modify/ set.	

Parameter Name	***Meaning***	***config file***	***Default***	***Possible values***
[ONLINE_TEST_CONTROL]DelayTask	Adds a CPU consumer task to avoid blocking of HCI.	hci.ini	Do not modify/ set.	
[ONLINE_TEST_CONTROL]ReceiverTaskSize	HCI message receiver task size in bytes.	hci.ini	Do not modify/ set.	
[ONLINE_TEST_CONTROL]Debug	Produces debug output when reading hci.ini.	hci.ini	Do not modify/ set.	
[ONLINE_TEST_CONTROL]EnditemListSize	Defines the number of enditems that can be stored in a message of type ENDITEMS_MESSAGE/APPEND_ENDITEMS (data distribution table/ announce enditems).	hci.ini	Do not modify/ set.	
[WINDOWS] PathnameStoredLength	Maximum n Defines the maumber of character for pathname in input text fields (e.g. name of the measurement in the Graph Facility Properties dialog).	hci.ini	200	Positive
[WINDOWS] PathnameDisplayLength	Defines the maximum number of character for pathname displayed in input text fields. If the pathname exceeds the display length scroll–buttons are added to the text field.	hci.ini	200	Positive
[COLOR] MonitoringDisabled MonitoringIn_Limits MonitoringSoft_Limit_Violation MonitoringDanger_Limit_Violation MonitoringUndefined	Decription of Color Specifications Colors can be specified by using color name defined in the color database (use Unix ”showrgb” to look up color database) or hexadecimal code. A hexadecimal code is specified as an initial sharp sign character followed by a hexadecimal specification in one of the following formats: #RGB (one character per color) #RRGGBB (two character per color) #RRRGGGBBB (three character per color) #RRRRGGGGBBBB (four character per color) where R, G, and B represent single haxadecimal digits (upper or lower case). When fewer than 16 bits each are specified, they represent the most significant bits of the value, For example, #3a7 is the same as #3000a0007000.	hci.ini	turquoise green yellow red turquoise	
[AP_STATUS] MaxWindows	Defines the maximum number of AP Status Displays that can be displayed in parallel.	hci.ini	4	1..Max-Windows-PerAp-plication

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
[AP_STATUS] UpdateRate	Defines how often the AP status window is updated (in seconds); the value 0 means it will be updated only if one or more of the values are changed.	hci.ini	5	0..10
[CLOCK] MaxWindows	Defines the maximum number of Clock windows that can be displayed in parallel.	hci.ini	4	1..Max-Windows-PerAp-plication
[CLOCK] ReplayFooter	Controls display of window footer when started in replay mode.	hci.ini	True, footer is displayed	Boolean
[CLOCK] UpdateRate	Defines how often the Clock window (replay mode only) is updated (in seconds); the value 0 means it will be updated only if one or more of the values are changed.	hci.ini	0	0..10
[CLOCK] SMT_Refresh	Refresh rate in seconds of SMT reading	hci.ini	0.333 seconds, i.e. the SMT is read every 1/3 second	Duration
[CLOCK] LocalTimeTCI	Time Code Identifier	hci.ini	LT	Text
[CLOCK] SMT_TCI	Time Code Identifier	hci.ini	SMT	Text
[COMMAND_FACILITY] MaxWindows	Defines the maximum number of Command Facilities that can be displayed in parallel.	hci.ini	4	1..Max-Windows-PerAp-plication
[COMMAND_FACILITY] History	Defines the size of the history buffer, i.e. the number of commands stored in the history.	hci.ini	200	Positive

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
[COMMAND_FACILITY] DontLog	Specifies a list of commands not to be logged (this is applicable for HLCL command windows, but not synoptic displays). The list may include: – procedure name for primary commands, – a reserved word (import, type) for commands that start with a reserved word, – a pathname for APs and command sequences from MDB,– a qualified name (pathname.identifier) for UCL library procedures, – a file name in string quotes for command sequences from files, – ”?” for the ? command, – ”:=” for assignments. – ”*” for all commands (to disable logging) LOG_SYNTAX_ERRORS en/disables logging for syntactically wrong commands. If logging is disabled with DONT_LOG = ”*” nothing will be logged even if LOG_SYNTAX_ERRORS is set.	hci.ini	? LIST	
[GRAPH_FACILITY] Default	Sets a default measurement name used when a Graph Facility comes up. This attribute should only be used if the pathname is used very often.	hci.ini	not set	Text
[GRAPH_FACILITY] MaxWindows	Defines the maximum number of Graph Facilities that can be displayed in parallel.	hci.ini	1	1..Max-Windows-PerAp-plication
[GRAPH_FACILITY] GraphName*N*	The attributes GraphNameN and GraphTemplateN define the graphs that can be used by the Graph Facility. In general, these attributes should not be modified.	hci.ini		
[GRAPH_FACILITY] GraphTemplate1*N*	The attributes GraphNameN and GraphTemplateN define the graphs that can be used by the Graph Facility. In general, these attributes should not be modified.	hci.ini		
[MAIN_MENU] EnableDisconnectedNodes	If on, disconnected nodes are selectable on the test node submenu.	hci.ini	Off	Boolean
[RAW_DATA_DUMP] MaxWindows	Maximum number of raw data dump tools that can be displayed in parallel.	hci.ini	1	1..Max-Windows-PerAp-plication
[RAW_DATA_DUMP] PacketDefault	Default packet pathname, displayed when the properties are poped up.	hci.ini	Should be omitted	Text

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
[RAW_DATA_DUMP] BaseDefault	Define the layout and behaviour of the properties dialog. Defines the base–button selected by default (button with base 16).	hci.ini	16	
[RAW_DATA_DUMP] BaseCount	Define the layout and behaviour of the properties dialog. Defines the number of base–buttons.	hci.ini	3 (default bases are # 16, 10, ASCII).	
[RAW_DATA_DUMP] Base*N*	Define the layout and behaviour of the properties dialog. BaseN define the base choosen after selection of base–button N, e.g. after # selection of base–button 2 the output is displayed in octal format. A base # between 128 and 255 is interpreted as ASCII.	hci.ini	2	
[RAW_DATA_DUMP] BytesPerLineDefault	Define the layout and behaviour of the properties dialog. Bytes–per–line–button selected by default.	hci.ini	32	
[RAW_DATA_DUMP] BytesPerLineCount	Define the layout and behaviour of the properties dialog. Number of bytes–per–line–buttons selected by default.	hci.ini	8	
[RAW_DATA_DUMP] BytesPerLine*N*	Define the layout and behaviour of the properties dialog. BytesPerLineN define the bytes per line choosen after selection of bytes–per–line–button N, e.g. selection of the third button will display 24 lines. Defaults are N multiplied with eight.	hci.ini	8	
[SAS_STATUS] MaxWindows	Defines the maximum number of SAS Status Displays that can be displayed in parallel.	hci.ini	4	1..Max-Windows-PerAp-plication
[SAS_STATUS] UpdateRate	Defines how often the AP status window is updated (in seconds); the value 0 means it will be updated only if one or more of the values are changed.	hci.ini	0	0..10
[SYNOPTIC_DISPLAY] MaxWindows	Defines the maximum number of Synoptic Displays that can be displayed in parallel.	hci.ini	8	1..Max-Windows-PerAp-plication

Parameter Name	***Meaning***	***config file***	***Default***	***Possible values***
[SYNOPTIC_DISPLAY] MinimumWindowSize	Minimum window size accepted for synoptic displays.	hci.ini	50	Positive
[SYNOPTIC_DISPLAY] MaxMenuItems	Maximum number of menu items of a popup menu in synoptics.	hci.ini	10	
[SYNOPTIC_DISPLAY] MaxVariables	Maximum number of variables that can be displayed in one synoptic	hci.ini	Should not be modified.	
[SYNOPTIC_DISPLAY] MaxPictureReplacer	Maximum number of picture replacements in parallel.	hci.ini	Should not be modified.	
[SYNOPTIC_DISPLAY] DisplayNotMaintainedItems	If enabled all items of a synoptic display that are not maintained by any test node are reported to the message handler.	hci.ini	Disable	
[SYNOPTIC_DISPLAY] NACQFlagBackgroundRed	Define the background color of the ’data acquisition’ flag as RGB value.	hci.ini	black	
[SYNOPTIC_DISPLAY] NACQFlagBackgroundGreen	Define the background color of the ’data acquisition’ flag as RGB value.	hci.ini	black	
[SYNOPTIC_DISPLAY] NACQFlagBackgroundBlue	Define the background color of the ’data acquisition’ flag as RGB value.	hci.ini	black	
[SYNOPTIC_DISPLAY] NACQFlagForegroundRed	Define the foreground color of the ’data acquisition’ flag as RGB value.	hci.ini	yellow	
[SYNOPTIC_DISPLAY] NACQFlagForegroundGreen	Define the foreground color of the ’data acquisition’ flag as RGB value.	hci.ini	yellow	
[SYNOPTIC_DISPLAY] NACQFlagForegroundBlue	Define the foreground color of the ’data acquisition’ flag as RGB value.	hci.ini	yellow	
[SYNOPTIC_DISPLAY] NACQFlagTextSize	Defines the text size of the ’data not acquired’ flag, 1 is very small, 9 very large.	hci.ini	2 (small)	
[SYNOPTIC_DISPLAY] NOT_MAINTAINEDFlagText	Text displayed on acquisition status flag	hci.ini	NMAINT	
[SYNOPTIC_DISPLAY] REQUESTEDFlagText	Text displayed on acquisition status flag	hci.ini	REQUESTED	
[SYNOPTIC_DISPLAY] NOT_ACQUIREDFlagText	Text displayed on acquisition status flag	hci.ini	NACQ	
[SYNOPTIC_DISPLAY] NOT_RECEIVEDFlagText	Text displayed on acquisition status flag	hci.ini	NRCD	
[SYNOPTIC_DISPLAY] INVALIDFlagText	Text displayed on acquisition status flag	hci.ini	INVAL	

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
[SYNOPTIC_DISPLAY] ACQUIREDFlagText	Text displayed on acquisition status flag	hci.ini	””	
[SYNOPTIC_DISPLAY] DATA_INTERRUPTIONFlag Text	Text displayed on acquisition status flag	hci.ini	INTERR	
[SYNOPTIC_DISPLAY] STATICFlagText	Text displayed on acquisition status flag	hci.ini	STATIC	
[SYNOPTIC_DISPLAY] NOT_MAINTAINEDPriority	If an output object has more than one measurement connected to it (e.g. a bar chart displaying \MEA_1 and \MEA_2) Synoptic Displays uses a priority list to determine the flag text, 0 is highest priority, 7 lowest (i.e. if \MEA_1 is NOT_ACQUIRED and \MEA_2 is STATIC, NOT_ACQUIRED will be displayed.	hci.ini	3	
[SYNOPTIC_DISPLAY] REQUESTEDPriority	If an output object has more than one measurement connected to it (e.g. a bar chart displaying \MEA_1 and \MEA_2) Synoptic Displays uses a priority list to determine the flag text, 0 is highest priority, 7 lowest (i.e. if \MEA_1 is NOT_ACQUIRED and \MEA_2 is STATIC, NOT_ACQUIRED will be displayed.	hci.ini	4	
[SYNOPTIC_DISPLAY] NOT_ACQUIREDPriority	If an output object has more than one measurement connected to it (e.g. a bar chart displaying \MEA_1 and \MEA_2) Synoptic Displays uses a priority list to determine the flag text, 0 is highest priority, 7 lowest (i.e. if \MEA_1 is NOT_ACQUIRED and \MEA_2 is STATIC, NOT_ACQUIRED will be displayed.	hci.ini	1	
[SYNOPTIC_DISPLAY] NOT_RECEIVEDPriority	If an output object has more than one measurement connected to it (e.g. a bar chart displaying \MEA_1 and \MEA_2) Synoptic Displays uses a priority list to determine the flag text, 0 is highest priority, 7 lowest (i.e. if \MEA_1 is NOT_ACQUIRED and \MEA_2 is STATIC, NOT_ACQUIRED will be displayed.	hci.ini	2	
[SYNOPTIC_DISPLAY] INVALIDPriority	If an output object has more than one measurement connected to it (e.g. a bar chart displaying \MEA_1 and \MEA_2) Synoptic Displays uses a priority list to determine the flag text, 0 is highest priority, 7 lowest (i.e. if \MEA_1 is NOT_ACQUIRED and \MEA_2 is STATIC, NOT_ACQUIRED will be displayed.	hci.ini	0	

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
[SYNOPTIC_DISPLAY] ACQUIREDPriority	If an output object has more than one measurement connected to it (e.g. a bar chart displaying \MEA_1 and \MEA_2) Synoptic Displays uses a priority list to determine the flag text, 0 is highest priority, 7 lowest (i.e. if \MEA_1 is NOT_ACQUIRED and \MEA_2 is STATIC, NOT_ACQUIRED will be displayed.	hci.ini	7	
[SYNOPTIC_DISPLAY] DATA_INTERRUPTIONPriority	If an output object has more than one measurement connected to it (e.g. a bar chart displaying \MEA_1 and \MEA_2) Synoptic Displays uses a priority list to determine the flag text, 0 is highest priority, 7 lowest (i.e. if \MEA_1 is NOT_ACQUIRED and \MEA_2 is STATIC, NOT_ACQUIRED will be displayed.	hci.ini	6	
[SYNOPTIC_DISPLAY] STATICPriority	If an output object has more than one measurement connected to it (e.g. a bar chart displaying \MEA_1 and \MEA_2) Synoptic Displays uses a priority list to determine the flag text, 0 is highest priority, 7 lowest (i.e. if \MEA_1 is NOT_ACQUIRED and \MEA_2 is STATIC, NOT_ACQUIRED will be displayed.	hci.ini	5	
[SYNOPTIC_DISPLAY] Stretching	If enabled, stretching makes the portion of the synoptic containing objects (graphs, input objects, static elements, etc.) exactly fit in the window. Stretching transforms the object's control points differently in the x and y dimensions. For that reason, stretching is automatically disabled for synoptics containing arcs and circles.	hci.ini	false	
[SYNOPTIC_DISPLAY] SameAspectRatio	If disabled, the portion of the synoptic containing objects fit in the window using a best fit algorithm to preserve the aspect ratio. In this case, either the top and bottom or the sides of the synoptic may not be visible. If enabled, the synoptic is drawn exactly as it appeared in GWDU.	hci.ini	false.	
[SYNOPTIC_DISPLAY] RedrawAreas	If enabled, all dynamic object areas are redrawn after update to show static objects on top of these. Such static objects (e.g. a line crossing a bar chart) are hidden by the update procedure.	hci.ini	disabled for performance reasons.	
[SYNOPTIC_DISPLAY] HelpWindowWidth	Default window width and size of the synoptic display help.	hci.ini	450	

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
[SYNOPTIC_DISPLAY] HelpWindowHeight	Default window width and size of the synoptic display help.	hci.ini	120	
[SYNOPTIC_DISPLAY] HelpWindowMemoryMaximum	Memory used by help window, should not be changed.	hci.ini	5_000	
[SYNOPTIC_DISPLAY] RestoreOriginalColorWhenNotMonitored	Restores original color of output element when monitoring is disabled. If false, Disabled* color is used	hci.ini	no	
[STATUS_DISPLAY] CCUItemWidth	Defines the size of the CCU item in character.	hci.ini	40	
[SYNOPTIC_DISPLAY] CCUItemNextRow	Defines if the CCU item shall be displayed on next row.	hci.ini	off	
[SYNOPTIC_DISPLAY] CCUItem	Defines whether the CCU item shall be shown	hci.ini	on	
[SYSTEM_ADVISORY] UpdateFrequency	Defines how often the System Advisory shall be updated in seconds.	hci.ini	5	
[SYSTEM_ADVISORY] CheckFrequency	Defines how often it shall be checked that the System Advisory was updated.	hci.ini	two times the update frequency	
[SYSTEM_ADVISORY] AcknowledgeService	Determines if the acknowledge service is toggled on or off.	hci.ini	on	
[SYSTEM_ADVISORY] AcknowledgeTimeOut	Sets when the acknowledge time out in seconds, i.e. the time after that Online Test Control will log that the alarm was not acknowledged.	hci.ini	10	
[SYSTEM_ADVISORY] BeepOnWarning	Enables beep on warnings (yellow color).	hci.ini	true	
[TEST_NODE_STATUS] MaxWindows	Defines the maximum number of Test Node Status Displays that can be displayed in parallel.	hci.ini	4	
[TEST_NODE_STATUS] AutoResize	Defines whether test node status automatically resizes the window after selection of a new group;	hci.ini	true	
[HCI_RPI] LoadSynopticTimeOut	Defines the time in milliseconds TES will wait until the load_synoptic library call is timed out,	hci.ini	20_000	
[HCI_RPI] Last_HCI_ApplicationId	Highest identifier that can be used for a HCI application.	hci.ini	32 (i.e. HCI_32)	

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
[HCI_RPI] ReceiverTaskSize	Size of the receiver task in bytes.	hci.ini	200_000 Must not be modified.	
[HCI_RPI] PollingInterval	Polling intervall (milliseconds) for network software message reception.	hci.ini	333 Must not be modified.	
[HELP] Online	Defines the name of the online help file.	hci.ini	vicos_hci_help Must not be modified	
[HELP] Path	Defines the path where the online help files are located.	hci.ini	Must not be modified.	
[HLCL] SequenceDirectoryVariable	Defines the location where to find the login/logout sequences.	hci.ini	user's home directory	
[HLCL] LoginSequenceFile	Defines the rest of the login/logout sequence file names, e.g. $HOME/.user/hlcl_login.seq.	hci.ini	"/.user/hlcl_login.seq"	
[LOG] LoadErrorMessages	Controls loading of error message definitions. If disabled error messages will neither be loaded from file nor from data base.	hci.ini	true.	
[LOG] LoadErrorMessagesFromDB	Controls loading error message definitions from data base. If disabled error messages are only loaded from file.	hci.ini	True	
[LOG] BufferSize	Defines the size of Online Test Control's internal log buffer.	hci.ini	100 should not be modified	
[LOG] Required	Defines whether Online Test Control requires logging services, i.e. if required and a connection to the TRDB is not possible it will terminate.	hci.ini	True.	
[LOG] Disabled	Disables logging totally.	hci.ini	false	

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
[LOG] CodeOffset	Offset added to the internal error codes of Online Test Control.	hci.ini	Should not be modified.	
[LOG] MessageFile	Defines the location of the message definition file.	hci.ini	$GSAF_HOME/hci/config/messages.def	
[LOG] UserName	.Defines the user name of the addressed message window.	hci.ini	all users ("" is same user as Online Test Control, "*" is all users).	
[SCREEN_SETUP] ConfigurationCheck	Defines whether a warning/confirmation is generated when a screen setup has to be loaded that was stored under a different configuration.	hci.ini	yes	
[SCREEN_SETUP] LineWidth	Defines the maximum line width/length in a screen setup file.	hci.ini	256	
[SCREEN_SETUP] ListRows	Defines the number of visible rows of the screen setup files list.	hci.ini	5	
[SCREEN_SETUP] Directory	Defines the location of the screen setup pool directory.	hci.ini	$HCI_HOME/data/screen_setup_pool	
[SMT_SIM] Offset	Offset to local time in seconds.	hci.ini	0.0	
[SMT_SIM] Available	Enables/disables SMT Simulator	hci.ini	true	
[TASKING] AP_INPUT_DIALOG.T_AP_INPUT_SENDER	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	128_000	

Parameter Name	*Meaning*	*config file*	*Default*	*Possible values*
[TASKING] AP_INPUT_DIALOG.T_AP_INPUT_OBCS	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	64_000	
[TASKING] APPLICATION_CONTROLLER.INTERIM_BUFFER_TASK_SIZE	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	20_000	
[TASKING] COMMAND_DISTRIBUTOR.T_OBCS_DELIVERY_NOTE	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	10_000	
[TASKING] COMMAND_DISTRIBUTOR.T_SERVER	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	100_000	
[TASKING] CMD_WINDOW_MANAGER	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	300_000	
[TASKING] DATA_DISTRIBUTOR.INTERIM_BUFFER_TASK_SIZE	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	48_000	
[TASKING] HCI_ENVIRONMENT	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	32_000	
[TASKING] HCI_SMT.T_SMT_READER	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	32_000	
[TASKING] LOGGING_SERVICE.INTERIM_BUFFER_TASK_SIZE	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	48_000	
[TASKING] ScreenSetupLoader	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	10_000	
[TASKING] SD_ANSWER_DIALOG.T_ANSWER_TASK	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	10_000	
[TASKING] SYNOPTIC_DISPLAYS_EXECUTOR.T_EXECUTOR	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	130_000	

Parameter Name	***Meaning***	***config file***	***Default***	***Possible values***
[TASKING] SYNOPTIC_DISPLAYS_EXECUTOR.T_PICTURE_REPLACER	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	130_000	
[TASKING] T_OBCS_DATA_DISTRIBUTOR	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	200_000	
[TASKING] T_HLCL_INTERPRETER	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	46_000	
[TASKING] T_INTERPOSER_TASK	Defines the internal attributes for tasking, must NOT be modified/set.	hci.ini	300_000	
[VALIDATION] Validation	Toggles validation on/off.	hci.ini	On	
[VALIDATION] Show	Controls display of the validation window.	hci.ini	window is shown (on)	
[VALIDATION] WindowWidth	Defines the size of the validation window.	hci.ini	400, 400	
[VALIDATION] WindowHeight	Defines the size of the validation window.	hci.ini	400, 400	
[VALIDATION] ErrorDiversion	Enables diversion of DataViews error messages to synoptic displays. If off, error messages are displayed on standard error (i.g. the console window).	hci.ini	on	
[VALIDATION] Logo	Logo file name, must be an absolute path without environment variables.	hci.ini	$GSAF_HOME/hci/config/logo.v	

L CGS SCREEN SETUP

L-1 Screen Setups and Window Definitions

There are several means to control and predefine the setup of the workstation's screens for CGS use.

The following save function are implemented separately:

Position and size of **Task Selector, Info, DB Selector, Welcome and Message Handler** can be saved by the Utilities –> Save_Workspace function of the window manager.

Message Handler window contents can be saved by storing the properties of the message window.

HCI Windows can be saved by the built–in Screen–Setup_Maintenance function

The screen definition files are stored under $HCI_HOME/data/screen_setup_pool.

For each user a specific setup can be defined in the file $CGS_HOME/config/USER_PROFILE. This screen setup is loaded ofter start of HCI. Standard Setup is defined in the file ”basic”.

TEV Windows cannot be saved.

A setup can, however, be defined in the $TEV_HOME/config/XDefaults file: For each TEV window (main windows and tool windows), the position of the window can be specified within this file. The file is attached to the X Resource database before TEV is started:

```
! Windows called from [Sessions] menu
! ————————————————————————————————————

! ”TEV : Test Evaluation” window
Tev.Mainmenu.x:                      195
Tev.Mainmenu.y:                      0

! ”TEV : Select Execution Sessions” window
Tev.ExecutionSession.main.x:         0
Tev.ExecutionSession.main.y:         25

! ”TEV : Execution Session List” window
Tev.ExecutionSession.sessionList.x:  210
Tev.ExecutionSession.sessionList.y:  0

! ”TEV : Evaluation Session” window
Tev.EvaluationSession.x:             95
Tev.EvaluationSession.y:             110
```

```
! Windows called from [Utilities] menu
! ————————————————————————————

! ”TEV : Working Directory Files” window
Tev.WorkingDirFiles.Main.x:              50
Tev.WorkingDirFiles.Main.y:              140

! ”TEV : Working Directory Files : Rename” window
Tev.WorkingDirFiles.Rename.x:            245
Tev.WorkingDirFiles.Rename.y:            270

! ”TEV : Working Directory Files : Copy” window
Tev.WorkingDirFiles.Copy.x:              245
Tev.WorkingDirFiles.Copy.y:              420


! Windows called from [Tools] menu
! ————————————————————————

! Four instances of each tool (n = 1..4)
! Windows <main_title>:
!    ”TEV : Events logging (n)”
!    ”TEV : Raw Data Dump (n)”
!    ”TEV : Data Set (n)”
!    ”TEV : Statistics Generation (n)”
!    ”TEV : Data Listing (n)”
!    ”TEV : Graph Display (n)”

! ”<main_title>” window
Tev*main1.x:          200
Tev*main1.y:          100
Tev*main2.x:          225
Tev*main2.y:          125
Tev*main3.x:          250
Tev*main3.y:          150
Tev*main4.x:          275
Tev*main4.y:          175

! ”<main_title> : <pop_up_title> ” window
!  with <pop_up_title>:
!          ”Definition Name”
!                ”Result File List”
!                ”Data Set Result”
Tev*popUpList1.x:     675
Tev*popUpList1.y:     100
Tev*popUpList2.x:     700
Tev*popUpList2.y:     125
Tev*popUpList3.x:     725
```

```
Tev*popUpList3.y:            150
Tev*popUpList4.x:            750
Tev*popUpList4.y:            175

! ”<main_title>” window
Tev*result1.x:               185
Tev*result1.y:               425
Tev*result2.x:               210
Tev*result2.y:               450
Tev*result3.x:               235
Tev*result3.y:               475
Tev*result4.x:               260
Tev*result4.y:               500

! ”<main_title> : Data Set Parameters” window
Tev*parameters1.x:           520
Tev*parameters1.y:           240
Tev*parameters2.x:           545
Tev*parameters2.y:           265
Tev*parameters3.x:           570
Tev*parameters3.y:           290
Tev*parameters4.x:           595
Tev*parameters4.y:           315

! ”TEV : Raw Data Dump (n)” window
Tev*RawData.packet1.x:                 450
Tev*RawData.packet1.y:                 300
Tev*RawData.packet2.x:                 475
Tev*RawData.packet2.y:                 325
Tev*RawData.packet3.x:                 500
Tev*RawData.packet3.y:                 350
Tev*RawData.packet4.x:                 525
Tev*RawData.packet4.y:                 375

! ”TEV : Raw Data Dump (n) : Output Format” window
Tev*RawData.outputFormat1.x:     690
Tev*RawData.outputFormat1.y:     100
Tev*RawData.outputFormat2.x:     715
Tev*RawData.outputFormat2.y:     125
Tev*RawData.outputFormat3.x:     740
Tev*RawData.outputFormat3.y:     150
Tev*RawData.outputFormat4.x:     765
Tev*RawData.outputFormat4.y:     175
```

TSCV Windows cannot be saved.

The size and position of the main window of TSCV can be given as parameters to the startup:The user may optionally specify by command line parameters the placement of the main TSCV window.

If the command line contains the tag **–geometry**, tscv will expect to find integer values for the width, height, horizontal postion, vertical position, respectively, formatted as follows: **wwwwxhhhhh+xxxx+yyyy**
The values for the size may be omitted, thus also the following format is legal: **+xxx+yyy**

To update the startup, the command line in $TSCV_HOME/bin/common/start_tscv can be adapted.

If geometry parameters are not provided on the command line, TSCV will search the X resource database. It will search for the resource variable **tscv.geometry**, and the variable string should be formatted as **wwwwxhhhhh+xxxx+yyyy**

As the X resource DB is updated during startup of the window manager by the file .Xdefaults, this file should contain the line

tscv.geometry: wwwwxhhhhh+xxxx+yyyy

CSS Windows: TBS

M UCL FILE IO VIA A SPECIFIC SAS

The UCL Language as used for CGS lacks definitions of operations to read from ordinary files resp. to write to ordinary files. Such operations are also not supported by UCL System Libraries. To overcome this situation a SAS has been developed which provides operations to Automated Procedures via a UCL User Library allowing reading or writing of ASCII strings from/to ordinary text files.

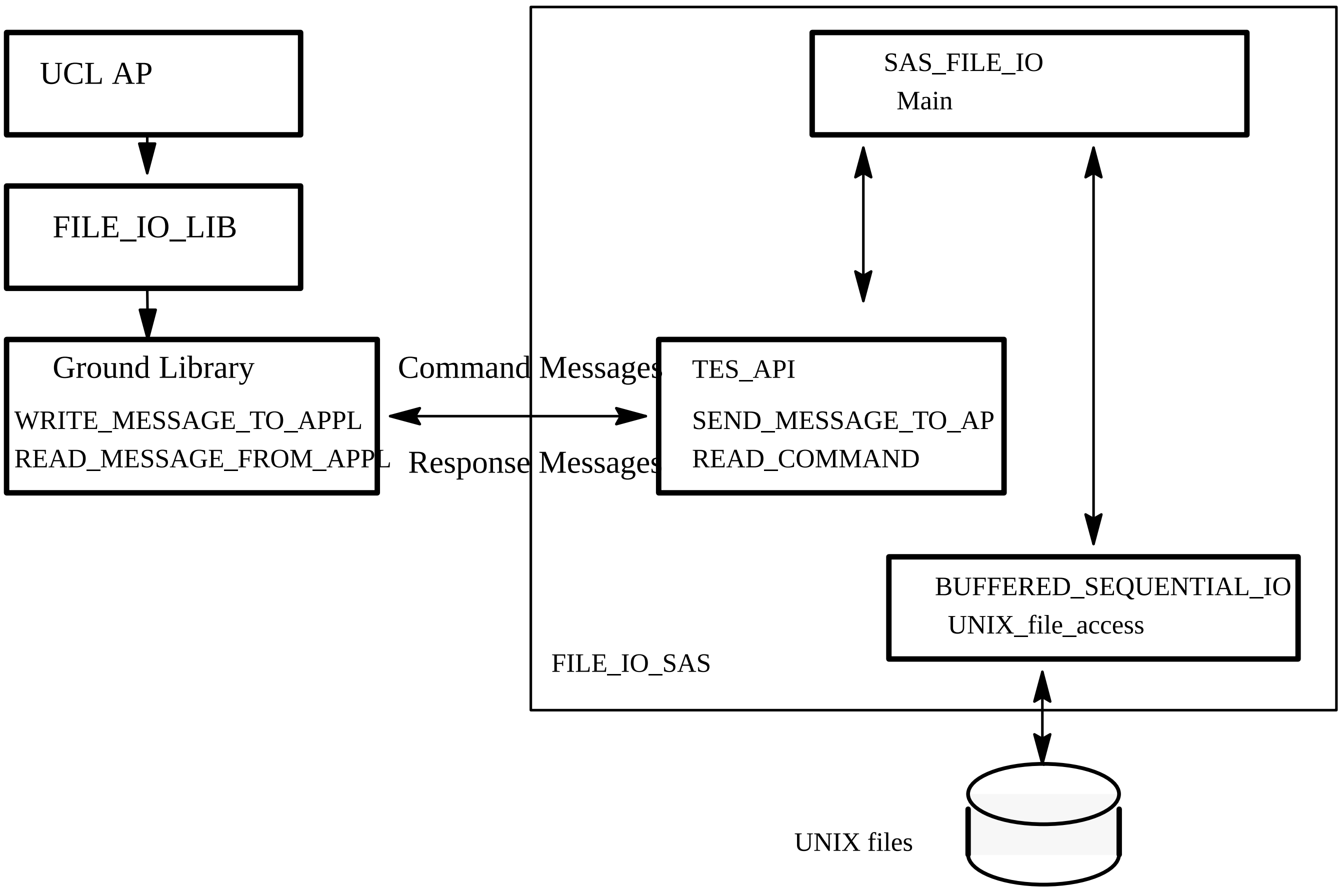
The communication between the AP and the SAS is implemented in a transparent way in the UCL User Library operations. Thus, from the user's point of view, the interface is very similar as if the operations would have been implemented in a UCL System Library.

REGISTER/UNREGISTER: Register the AP to the SAS
OPEN/CLOSE: Open a file for input or output resp close the file.
GET, GETLINE: Read text from a file
PUT,NEWLINE: Write text to a file

END_OF_FILE, END_OF_LINE, SKIP, SKIP_LINE, HANDLE_VALID: Status and control operations

To allow for deletion of files, the **DELETE** operation is available.

There is an additional operation implemented allowing execution of any UNIX command from within UCL:
EXECUTE: Execute a UNIX command via "/bin/sh"



M-1 Communication between FILE_IO_LIB and SAS_FILE_IO

The procedures from FILE_IO_LIB and SAS_FILE_IO communicate by exchanging messages. Messages are simple ASCII–strings. In principle the communication between an AP which uses FILE_IO_LIB and SAS_FILE_IO is performed as follows:

- The FILE_IO_LIB procedure creates a command message which is sent to SAS_FILE_IO using the UCL function WRITE_MESSAGE_TO_APPLICATION. It then starts waiting for the response message.
- SAS_FILE_IO reads the command message and sends an acknowledge to CGS using ACKNOWLEDGE_COMMAND.
- The command message is stored in a command buffer until SAS_FILE_IO is able to process it.
- The message is parsed and the appropriate actions are executed.
- A response message containing the result of command execution is sent to the AP with the command SEND_MESSAGE_TO_AP.
- FILE_IO_LIB procedure reads the response message using READ_MESSAGE_FROM_APPLICATION and evaluates it.

For realization of this basic communication scheme the following assumptions must be met:

1. Before the AP can send a message, it has to know the Application–ID of SAS_FILE_IO. This ID is stored in the housekeeping variable SAS_ID. The full pathname of SAS_ID is specified within the body of FILE_IO_LIB.
2. SAS_FILE_IO must know the full AP–name, which is expected as a parameter value for SEND_MESSAGE_TO_AP. The AP has to inform the SAS of it’s name within the first message.
3. Several instances of an AP can be connected simultaneously with SAS_FILE_IO. If more than one instance of the AP is running, a message is passed to all instances. Therefore, the AP–name is not sufficient for identification of the receiver (or sender) of a message. Every message contains an unique signature, which can be assigned to exactly one AP instance. Messages which do not contain the correct signature can safely be ignored by the AP instance. As signature first the AP–ID of the AP instance and later an unique AP–handle is used.

After determination of the Application–ID of SAS_FILE_IO the AP sends a message with it’s full name to SAS_FILE_IO, which is signed with the AP–ID. SAS_FILE_IO stores the name of the AP in an internal table and assigns a unique AP–handle to the AP. The AP–handle is sent the AP in the response message, which is signed with the AP–ID. All future messages from and to this specific instance of the AP are signed with the delivered AP–handle.

During buffering and processing a command message, SAS_FILE_IO sends approximately all ?? seconds a contact message to the sender of the command. If FILE_IO_LIB does not receive either the response message or a contact message within a predefined number of seconds after sending the command message or receiving a contact message, it stops waiting and returns an connection error.

M-1.1 Messages

As already mentioned above, messages are simple ASCII strings. For separation of the various components the character ’;’ is used as delimiter.

Command messages

Command messages are sent from an FILE_IO_LIB procedure to SAS_FILE_IO. They can be up to 255 characters long. The format of a command message is as follows:

keyword ; *AP–handle* ; *parameters*

keyword identifies the desired operation. It usually corresponds to the name of the calling FILE_IO_LIB procedure. *AP–handle* is the handle which is assigned to the AP–instance. The contents and format of *parameters* depend on the operation to be executed. Therefore, *parameters* is evaluated depending on *keyword*.

Response messages

Response messages are sent from SAS_FILE_IO to the AP. They are limited of 80 characters. The format of a response message is:

AP–handle ; *message type* ; *result*

AP–handle is the handle of the AP which sent the respective command message. *message type* is a single character:

- *message type* = ’0’ means the command executes successful (result message). In this case *result* contains the results. Format and contents of *result* depends on the performed operation.
- *message type* = ’1’ announces that an error has occurred (error message). In this case *result* contains the error message.
- *message type* = ’2’ means that SAS_FILE_IO was not yet able to process the command (contact message). In this case *result* is empty.

M-1.2 Error handling

Errors might be detected both in SAS_FILE_IO and FILE_IO_LIB. SAS_FILE_IO usually informs the user not directly about an occurred error. Instead of this, SAS_FILE_IO attempts to send the error message to the AP instance which sent the appropriate command message. Sometimes this is not possible, e.g. if the sender of the command message can not be determined. In this case the error message is passed to CGS using SEND_ERROR_MESSAGE.

If FILE_IO_LIB receives an error message or detects an error, it delivers the error message to the user using WRITE_MESSAGE_TO_USER. The procedure then returns the respective error status as result code.

M-1.3 Procedures in File_IO_Lib

Each call to a procedure from FILE_IO_LIB returns a result code. Codes which can be returned by all or nearly all procedures are:

- OKAY — everything o.k.
- CONNECTION_ERROR — The connection to SAS_FILE_IO is broken.
- NOT_REGISTERED — The AP is not connected to SAS_FILE_IO.
- INVALID_FILE_HANDLE — The handle has not been associated with a file.

Some result codes should never appear under normal circumstances and indicate most likely a bug in FILE_IO_LIB or SAS_FILE_IO:

- BUFFER_OVERFLOW — The command buffer in SAS_FILE_IO is full.
- MESSAGE_FORMAT_ERROR — SAS_FILE_IO could not parse the command message. This should never happen. May be FILE_IO_LIB or SAS_FILE_IO is obsolete.
- DEVICE_ERROR — a problem with the hardware
- END_ERROR
- DATA_ERROR
- OTHER_ERROR

M-1.3.1 REGISTERED

- **Syntax:** function REGISTERED : boolean;
- **Parameters:** none
- **Notes:** This function returns true, if the AP is connected to SAS_FILE_IO, otherwise false.

M-1.3.2 REGISTER

- **Syntax:**
```
procedure REGISTER ( in AP_NAME       : string;
                     in HOST          : NODE_NAME;
                     out FIO_RESULT   : FIO_RETURN);
```
- **Parameters:**
  - AP_NAME is the pathname of the AP.
  - HOST specifies the host, on which SAS_FILE_IO will be started if it is not already running.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

COMMAND_TOO_LONG	The length of the AP_NAME together with the other message components exceeds the maximal length of a command message.
ALREADY_REGISTERED	REGISTER was already called before.
REGISTER_REJECTED	There is no free AP–handle.

- **Notes:** REGISTER has to be called before using other FILE_IO_LIB procedures. It establishes the connection to SAS_FILE_IO.

M-1.3.3 UNREGISTER

- **Syntax:** procedure UNREGISTER (out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - FIO_RESULT is the result code returned by this call.
- **Notes:** This procedure disconnects the AP from SAS_FILE_IO. All files, which are still opened by this AP, are closed before. SAS_FILE_IO not checks whether a registered AP still exists. It is highly recommended to call UNREGISTER before finishing the AP. Otherwise the AP–handle is permanently locked and can be released only by resetting or restarting SAS_FILE_IO.

M-1.3.4 OPEN

- **Syntax:**
```
procedure OPEN ( in FILENAME       : string;
                 in MODE           : FILE_MODE;
                 out HANDLE        : FILE_HANDLE;
                 out FIO_RESULT    : FIO_RETURN);
```
- **Parameters:**
  - FILENAME consists of the path and the name of the file to be opened.
  - MODE specifies the allowed operations for the file.

IN_FILE	opens an existing file for reading. All input (GET, GET_LINE) and status operations (END_OF_LINE, END_OF_FILE) are allowed.
OUT_FILE	creates a new file. If the file already exists, it is truncated to zero length. Only output operations (PUT, NEW_LINE, PUT_LINE) are allowed.
APPEND_FILE	opens an existing file for appending new data. Only output operations are allowed.

  - HANDLE is an unique file handle to be used in future calls to FILE_IO_LIB procedures.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

COMMAND_TOO_LONG	The length of the AP_NAME together with the other message components exceeds the maximal length of a command message.
TOO_MANY_OPEN_FILES	All file handles are already assigned.
FILE_LOCKED	The file is locked (see below).
NAME_ERROR	FILENAME parameter is invalid.
USE_ERROR	It is impossible to open resp. create a file with the specified name.

- **Notes:** A file can be opened simultaneously several times for reading. It is not possible to open a file, which is already opened for writing, or to open a file for writing, which is already open.

M-1.3.5 CLOSE

- **Syntax:** procedure CLOSE (in HANDLE : FILE_HANDLE;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - FIO_RESULT is the result code returned by this call.
- **Notes:** none

M-1.3.6 DELETE

- **Syntax:** procedure DELETE (in FILENAME : string;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - FILENAME is name of the file to be deleted.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

COMMAND_TOO_LONG	The length of FILENAME together with the other message components exceeds the maximal length of a command message.
FILE_LOCKED	The file is open.
NAME_ERROR	FILENAME parameter is invalid.
USE_ERROR	Deletion of the specified file is not possible.

- **Notes:** It’s not possible to delete an open file.

M-1.3.7 HANDLE_VALID

- **Syntax:** procedure HANDLE_VALID (in HANDLE : FILE_HANDLE;
  out RESULT : boolean;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - RESULT is true if HANDLE is assigned to an open file, otherwise false.
  - FIO_RESULT is the result code returned by this call.
- **Notes:** none

M-1.3.8 END_OF_LINE

- **Syntax:** procedure END_OF_LINE (in HANDLE : FILE_HANDLE;
  out RESULT : boolean;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - RESULT is true if the last read character is a line feed or the last character in the file has been read.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

MODE_ERROR	The file was opened for output.
- **Notes:** This procedure is only applicable for input files.

M-1.3.9 END_OF_FILE

- **Syntax:** procedure END_OF_FILE (in HANDLE : FILE_HANDLE;
  out RESULT : boolean;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - RESULT is true if the last character in the file has been read.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

MODE_ERROR	The file was opened for output.
- **Notes:** This procedure is only applicable for input files.

M-1.3.10 PUT

- **Syntax:** procedure PUT (in HANDLE : FILE_HANDLE;
  in ITEM : string;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - ITEM contains the characters which will be appended to the file.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

MODE_ERROR	The file was opened for input.
- **Notes:** This procedure is only applicable for output files. ITEM can include all kinds of characters.

M-1.3.11 GET

- **Syntax:** procedure GET(in HANDLE : FILE_HANDLE;
  out ITEM : string;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - ITEM contains the characters read from file.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

MODE_ERROR	The file was opened for output.
- **Notes:** This procedure reads characters from the input file into ITEM until either ITEM is full or the end of the file is reached. The characters are not interpreted anyway. In case of an attempt to read past the end of the file ITEM is empty.

M-1.3.12 SKIP

- **Syntax:** procedure SKIP(in HANDLE : FILE_HANDLE;
  in CHARS_TO_SKIP : unsigned_integer;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - CHARS_TO_SKIP is the number of characters to be ignored in the input file.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

MODE_ERROR	The file was opened for output.
- **Notes:** This procedure advances the input position over CHARS_TO_SKIP characters.

M-1.3.13 NEW_LINE

- **Syntax:** procedure NEW_LINE (in HANDLE : FILE_HANDLE;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

MODE_ERROR	The file was opened for input.
- **Notes:** This procedure appends a line terminator (line feed) to the file.

M-1.3.14 PUT_LINE

- **Syntax:** procedure PUT_LINE (in HANDLE : FILE_HANDLE;
  in ITEM : string;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - ITEM contains the characters which will be appended to the file.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

MODE_ERROR	The file was opened for input.

- **Notes:** A call on PUT_LINE is equivalent to a call on PUT followed by a call on NEW_LINE.

M-1.3.15 GET_LINE

- **Syntax:** procedure GET_LINE (in HANDLE : FILE_HANDLE;
  out ITEM : string;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - ITEM contains the characters read from file.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

MODE_ERROR	The file was opened for output.

- **Notes:** This procedure fetches characters from the input file into ITEM until either ITEM is full or the end of the line is reached. The line terminator (line feed) itself not appears in ITEM. The other characters are not interpreted anyway. In case of an attempt to read past the end of the file ITEM is empty.

M-1.3.16 SKIP_LINE

- **Syntax:** procedure SKIP_LINE (in HANDLE : FILE_HANDLE;
  in LINES_TO_SKIP : unsigned_integer;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - LINES_TO_SKIP is the number of lines to be ignored in the input file.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

MODE_ERROR	The file was opened for output.

- **Notes:** This procedure advances to the beginning of the LINES_TO_SKIP'th following line . If there aren't so much lines left, it advances the input position to the end of the file.

M-1.3.17 EXECUTE

- **Syntax:** procedure EXECUTE (in COMMAND : string;
  out FIO_RESULT : FIO_RETURN);

- **Parameters:**
  - COMMAND is the command string to be executed.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

COMMAND_TOO_LONG	The length of the COMMAND together with the other message components exceeds the maximal length of a command message.
PROCESS_ERROR	No new process could be created.

- **Notes:** SAS_FILE_IO creates a new process and executes COMMAND using ’/bin/sh’. The created process is completely independent from SAS_FILE_IO. It is up to the user to redirect input and output if necessary. FIO_RESULT contains only information about message delivery (to SAS_FILE_IO) and process creation, but not about the created process.

M-1.3.18 ADD_FILE_TO_TEST_SESSION

- **Syntax:** procedure ADD_FILE_TO_TEST_SESSION

```
(in FILE_NAME    : STRING; — full UNIX pathname
 in SESSION      : STRING; — name of test session
 in PRODUCER     : STRING; — name identifying the producer of the file
 in CREATION_TIME: TIME ;   — time when file was created
 out FIO_RESULT         : FIO_RETURN);

— Store a file in the specified Test Execution Session within the Test Result
— Database(TRDB). File is transfered to the TRDB directory and maintained as
— part of the test session
— Note: To get the currently open test session, the user is obliged to read
—      the name from a SW Variable / HK value before calling this operation
```

- **Parameters:**
  - FILE_NAME is the name of the file to be added (full pathname)
  - SESSION is the name of the test session
  - PRODUCER is the name software item having created the file
  - CREATION_TIME: time when file was created
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

SESSION_ERROR	Session is not defined
PROCESS_ERROR	could not create a new process to transfer/register the file

M-2 Installation of SAS_FILE_IO and FILE_IO_LIB

The following instructions describe one way to integrate FILE_IO_LIB into an existing CGS system.

1. **Installation of SAS_FILE_IO**
   - Copy or link SAS_FILE_IO executable to directory $SAS_HOME/bin.
   - Enter the SAS name (normally SAS_FILE_IO), the port number and the host, on which SAS_FILE_IO will be executed, in the SYSTEM_TOPOLOGY_TABLE.

     Note: Verify, that on the node where the SAS_FILE_IO is assigned to, **no TES is assigned** as well. The SAS first looks on a locally assigned TES to connect to, and thus the connection to another test node (TES) having started the SAS would fail. Thus, every other SUN in the topology table is possible, but no other SUN being a test node.
   - Create an end item of type EGSE_SOFTWARE in the MDB and enter the required information(SAS_TYPE,SHORT_NAME). The short name defined for the SAS in this item has to correspond exactly to the name of the SAS executable file.
2. **Installation of FILE_IO_LIB**
   - Create an end item of type UCL_USER_LIBRARY and copy specification and body of FILE_IO_LIB into the created library.
   - Setup the library specification:
     - Check the pathname of the imported GROUND_LIBRARY.
   - Setup the library body:
     - Check the pathname of the imported GROUND_LIBRARY and SUPPORT_LIBRARY.
   - Compile and store library specification and body.
3. **Setup the test configuration**
   - The host on which SAS_FILE_IO will be executed has to participate on the test. Insert the appropriate entry in menu **Open–>EGSE Workstation Nodes** resp. in menu **Open–>EGSE Test Nodes** or **Open–>DB Server Node**
   - Ensure that the CDU, which holds the items for the software variable and the SAS, is loaded by a test node (check in menu **View–>EGSE Test Node Items**).
   - Assign the SAS to a test node in menu **View–>EGSE Test Node SASs**. Note that this entry has to correspond to entry in SYSTEM_TOPOLOGY_TABLE in the body of FILE_IO_LIB.
   - Generate a new Scoe file for the Test Configuration by selecting **Tools–>Generate Scoe Files**.
4. **Shutdown and restart CGS**,
   in order to let the modifications in SYSTEM_TOPOLOGY_TABLE and the generated Scoe files become effective.

N MDB CONSISTENCY CHECKS

The MDB Consistency Checker Program can be called from the I_MDB program for the scopes CDU or CCU. It performes predefined checks on all the enditems defined in that scope.

There is another program for specific enditems that can be called for single enditems. It performs checks on the data defined for this enditems. References to other enditems are checked within the current scope (CCU or CDU).

N-1 List of CGS Standard Consistency Checks

The identifiers “CC– ...” are the consistency checker output messages.
Variable message parts are shown as “[...]”.

N-1.1 Mandatory Checks

CC–1 Mandatory aggregate is not defined

The following aggregates are mandatory:

```
type name                        aggregate name
----------------------------------------------------------------------------------------------------------
APID                             APID Table
BOOLEAN_MEASUREMENT              Boolean Calibration
BOOLEAN_MEASUREMENT              Integer Raw Value Range            (if Raw Value Type.Raw Value Type
                                                                        = SIGNED_INTEGER)
BOOLEAN_MEASUREMENT              Raw Value Size in Bits             (if Raw Value Type.Raw Value Type
                                                                        = SIGNED_INTEGER)
BOOLEAN_MEASUREMENT              Raw Value Type
BOOLEAN_STIMULUS                 Boolean Calibration
BOOLEAN_STIMULUS                 Formal Parameter
BOOLEAN_STIMULUS                 Integer Raw Value Range            (if Raw Value Type.Raw Value Type
                                                                        = SIGNED_INTEGER)
BOOLEAN_STIMULUS                 Raw Value Size in Bits             (if Raw Value Type.Raw Value Type
                                                                        = SIGNED_INTEGER)
BOOLEAN_STIMULUS                 Raw Value Type
BOOLEAN_SW_VARIABLE              Initial Boolean Value
BURST_PULSE_STIMULUS             Analog Decalibration Coefficients  (if Calib Curve Type.Curve Type
                                                                        = POLYNOM)
BURST_PULSE_STIMULUS             Analog Decalibration Point Pairs   (if Calib Curve Type.Curve Type
                                                                        = POINT_PAIRS)
BURST_PULSE_STIMULUS             Calib Curve Type
BURST_PULSE_STIMULUS             Raw Value Size in Bits             (if Raw Value Type.Raw Value Type
                                                                        = UNSIGNED_INTEGER)
BURST_PULSE_STIMULUS             Raw Value Type
BURST_PULSE_STIMULUS             Unsigned Integer Engineering Range
BURST_PULSE_STIMULUS             Unsigned Integer Raw Value Range   (if Raw Value Type.Raw Value Type
                                                                        = UNSIGNED_INTEGER)
CCSDS_ADU_DESCRIPTION            ADU General Info
CCSDS_ADU_DESCRIPTION            CCSDS Primary Header
CCSDS_ADU_DESCRIPTION            Measurement End Items
CPL_SCRIPT                       Code
CPL_SCRIPT                       Compilation Date
CPL_SCRIPT                       Source
DEMO_ANALOG_STIMULUS             EGSE Stimulus General Info
DEMO_ANALOG_STIMULUS             Formal Parameter
DEMO_BOOLEAN_MEASUREMENT         Boolean Calibration
DEMO_BOOLEAN_MEASUREMENT         Integer Raw Value Range            (if Raw Value Type.Raw Value Type
                                                                        = SIGNED_INTEGER)
DEMO_BOOLEAN_MEASUREMENT         Raw Value Size in Bits             (if Raw Value Type.Raw Value Type
                                                                        = SIGNED_INTEGER)
DEMO_BOOLEAN_MEASUREMENT         Raw Value Type
DEMO_BOOLEAN_SW_VARIABLE         Initial Boolean Value
DEMO_BYTE_STREAM_MEASUREMENT     Bytestream Calibration
DEMO_BYTE_STREAM_MEASUREMENT     Raw Value Size in Bytes            (if Raw Value Type.Raw Value Type
                                                                        = BYTESTREAM)
DEMO_BYTE_STREAM_MEASUREMENT     Raw Value Type
DEMO_DISCRETE_MEASUREMENT        Discrete Calibration
DEMO_DISCRETE_MEASUREMENT        Raw Value Size in Bits             (if Raw Value Type.Raw Value Type
                                                                        = UNSIGNED_INTEGER)
DEMO_DISCRETE_MEASUREMENT        Raw Value Type
DEMO_DISCRETE_MEASUREMENT        Unsigned Integer Raw Value Range   (if Raw Value Type.Raw Value Type
                                                                        = UNSIGNED_INTEGER)
DEMO_DISCRETE_STIMULUS           Discrete Calibration
DEMO_DISCRETE_STIMULUS           EGSE Stimulus General Info
DEMO_DISCRETE_STIMULUS           Formal Parameter
DEMO_D_FLOAT_MEASUREMENT         Analog Calibration Coefficients    (if Calib Curve Type.Curve Type
```

```
                                                                      = POLYNOM)
DEMO_D_FLOAT_MEASUREMENT        Analog Point Pairs                  (if Calib Curve Type.Curve Type
                                                                        = POINT_PAIRS)
DEMO_D_FLOAT_MEASUREMENT        Calib Curve Type
DEMO_D_FLOAT_MEASUREMENT        Double Float Engineering Range
DEMO_D_FLOAT_MEASUREMENT        Integer Raw Value Range             (if Raw Value Type.Raw Value Type
                                                                        = SIGNED_INTEGER)
DEMO_D_FLOAT_MEASUREMENT        Raw Value Size in Bits              (if Raw Value Type.Raw Value Type
                                                                        in (SIGNED_INTEGER,
                                                                            UNSIGNED_INTEGER))
DEMO_D_FLOAT_MEASUREMENT        Raw Value Type
DEMO_D_FLOAT_MEASUREMENT        Unsigned Integer Raw Value Range    (if Raw Value Type.Raw Value Type
                                                                        = UNSIGNED_INTEGER)
DEMO_D_FLOAT_SW_VARIABLE        Double Float Engineering Range
DEMO_D_FLOAT_SW_VARIABLE        Initial Double Float Value
DEMO_FLOAT_MEASUREMENT          Analog Calibration Coefficients     (if Calib Curve Type.Curve Type
                                                                        = POLYNOM)
DEMO_FLOAT_MEASUREMENT          Analog Point Pairs                  (if Calib Curve Type.Curve Type
                                                                        = POINT_PAIRS)
DEMO_FLOAT_MEASUREMENT          Calib Curve Type
DEMO_FLOAT_MEASUREMENT          Float Raw Value Range               (if Raw Value Type.Raw Value Type
                                                                        = FLOAT)
DEMO_FLOAT_MEASUREMENT          Integer Raw Value Range             (if Raw Value Type.Raw Value Type
                                                                        = SIGNED_INTEGER)
DEMO_FLOAT_MEASUREMENT          Raw Value Size in Bits              (if Raw Value Type.Raw Value Type
                                                                        in (FLOAT,
                                                                            SIGNED_INTEGER,
                                                                            UNSIGNED_INTEGER))
DEMO_FLOAT_MEASUREMENT          Raw Value Type
DEMO_FLOAT_MEASUREMENT          Unsigned Integer Raw Value Range    (if Raw Value Type.Raw Value Type
                                                                        = UNSIGNED_INTEGER)
DEMO_INTEGER_MEASUREMENT        Analog Calibration Coefficients     (if Calib Curve Type.Curve Type
                                                                        = POLYNOM)
DEMO_INTEGER_MEASUREMENT        Analog Point Pairs                  (if Calib Curve Type.Curve Type
                                                                        = POINT_PAIRS)
DEMO_INTEGER_MEASUREMENT        Calib Curve Type
DEMO_INTEGER_MEASUREMENT        Integer Raw Value Range             (if Raw Value Type.Raw Value Type
                                                                        = SIGNED_INTEGER)
DEMO_INTEGER_MEASUREMENT        Raw Value Size in Bits              (if Raw Value Type.Raw Value Type
                                                                        in (SIGNED_INTEGER,
                                                                            UNSIGNED_INTEGER))
DEMO_INTEGER_MEASUREMENT        Raw Value Type
DEMO_INTEGER_MEASUREMENT        Unsigned Integer Raw Value Range    (if Raw Value Type.Raw Value Type
                                                                        = UNSIGNED_INTEGER)
DEMO_TM_PACKET                  ADU General Info
DEMO_TM_PACKET                  CCSDS Primary Header
DEMO_TM_PACKET                  Global Length
DEMO_TM_PACKET                  Measurement End Items
DEMO_U_INT_MEASUREMENT          Analog Calibration Coefficients     (if Calib Curve Type.Curve Type
                                                                        = POLYNOM)
DEMO_U_INT_MEASUREMENT          Analog Point Pairs                  (if Calib Curve Type.Curve Type
                                                                        = POINT_PAIRS)
DEMO_U_INT_MEASUREMENT          Calib Curve Type
DEMO_U_INT_MEASUREMENT          Raw Value Size in Bits              (if Raw Value Type.Raw Value Type
                                                                        = UNSIGNED_INTEGER)
DEMO_U_INT_MEASUREMENT          Raw Value Type
DEMO_U_INT_MEASUREMENT          Unsigned Integer Engineering Range
DEMO_U_INT_MEASUREMENT          Unsigned Integer Raw Value Range    (if Raw Value Type.Raw Value Type
                                                                        = UNSIGNED_INTEGER)
DEMO_U_INT_SW_VARIABLE          Initial Unsigned Integer Value
DEMO_U_INT_SW_VARIABLE          Unsigned Integer Engineering Range
DOUBLE_FLOAT_MEASUREMENT        Analog Calibration Coefficients     (if Calib Curve Type.Curve Type
                                                                        = POLYNOM)
DOUBLE_FLOAT_MEASUREMENT        Analog Point Pairs                  (if Calib Curve Type.Curve Type
                                                                        = POINT_PAIRS)
DOUBLE_FLOAT_MEASUREMENT        Calib Curve Type
DOUBLE_FLOAT_MEASUREMENT        Double Float Engineering Range
DOUBLE_FLOAT_MEASUREMENT        Integer Raw Value Range             (if Raw Value Type.Raw Value Type
                                                                        = SIGNED_INTEGER)
DOUBLE_FLOAT_MEASUREMENT        Raw Value Size in Bits              (if Raw Value Type.Raw Value Type
                                                                        in (SIGNED_INTEGER,
                                                                            UNSIGNED_INTEGER))
```

DOUBLE_FLOAT_MEASUREMENT	Raw Value Type	
DOUBLE_FLOAT_MEASUREMENT	Unsigned Integer Raw Value Range	(if Raw Value Type.Raw Value Type = UNSIGNED_INTEGER)
DOUBLE_FLOAT_STIMULUS	Analog Decalibration Coefficients	(if Calib Curve Type.Curve Type = POLYNOM)
DOUBLE_FLOAT_STIMULUS	Analog Decalibration Point Pairs	(if Calib Curve Type.Curve Type = POINT_PAIRS)
DOUBLE_FLOAT_STIMULUS	Calib Curve Type	
DOUBLE_FLOAT_STIMULUS	Double Float Engineering Range	
DOUBLE_FLOAT_STIMULUS	Formal Parameter	
DOUBLE_FLOAT_STIMULUS	Integer Raw Value Range	(if Raw Value Type.Raw Value Type = SIGNED_INTEGER)
DOUBLE_FLOAT_STIMULUS	Raw Value Size in Bits	(if Raw Value Type.Raw Value Type in (SIGNED_INTEGER, UNSIGNED_INTEGER))
DOUBLE_FLOAT_STIMULUS	Raw Value Type	
DOUBLE_FLOAT_STIMULUS	Unsigned Integer Raw Value Range	(if Raw Value Type.Raw Value Type = UNSIGNED_INTEGER)
DOUBLE_FLOAT_SW_VARIABLE	Double Float Engineering Range	
DOUBLE_FLOAT_SW_VARIABLE	Initial Double Float Value	
EGSE_ANALOG_STIMULUS	EGSE Stimulus General Info	
EGSE_ANALOG_STIMULUS	Formal Parameter	
EGSE_BINARY_PACKET	EGSE Stimulus General Info	
EGSE_BYTE_STREAM_MEASUREMENT	Bytestream Calibration	
EGSE_BYTE_STREAM_MEASUREMENT	Raw Value Size in Bytes	(if Raw Value Type.Raw Value Type = BYTESTREAM)
EGSE_BYTE_STREAM_MEASUREMENT	Raw Value Type	
EGSE_DISCRETE_DERIVED_VALUE	Compilation Date	
EGSE_DISCRETE_DERIVED_VALUE	Expression Code	
EGSE_DISCRETE_DERIVED_VALUE	Expression Source	
EGSE_DISCRETE_MEASUREMENT	Discrete Calibration	
EGSE_DISCRETE_MEASUREMENT	Raw Value Size in Bits	(if Raw Value Type.Raw Value Type = UNSIGNED_INTEGER)
EGSE_DISCRETE_MEASUREMENT	Raw Value Type	
EGSE_DISCRETE_MEASUREMENT	Unsigned Integer Raw Value Range	(if Raw Value Type.Raw Value Type = UNSIGNED_INTEGER)
EGSE_DISCRETE_STIMULUS	Discrete Calibration	
EGSE_DISCRETE_STIMULUS	EGSE Stimulus General Info	
EGSE_DISCRETE_STIMULUS	Formal Parameter	
EGSE_FLOAT_DERIVED_VALUE	Compilation Date	
EGSE_FLOAT_DERIVED_VALUE	Expression Code	
EGSE_FLOAT_DERIVED_VALUE	Expression Source	
EGSE_FLOAT_MEASUREMENT	Analog Calibration Coefficients	(if Calib Curve Type.Curve Type = POLYNOM)
EGSE_FLOAT_MEASUREMENT	Analog Point Pairs	(if Calib Curve Type.Curve Type = POINT_PAIRS)
EGSE_FLOAT_MEASUREMENT	Calib Curve Type	
EGSE_FLOAT_MEASUREMENT	Float Raw Value Range	(if Raw Value Type.Raw Value Type = FLOAT)
EGSE_FLOAT_MEASUREMENT	Integer Raw Value Range	(if Raw Value Type.Raw Value Type = SIGNED_INTEGER)
EGSE_FLOAT_MEASUREMENT	Raw Value Size in Bits	(if Raw Value Type.Raw Value Type in (FLOAT, SIGNED_INTEGER, UNSIGNED_INTEGER))
EGSE_FLOAT_MEASUREMENT	Raw Value Type	
EGSE_FLOAT_MEASUREMENT	Unsigned Integer Raw Value Range	(if Raw Value Type.Raw Value Type = UNSIGNED_INTEGER)
EGSE_INTEGER_DERIVED_VALUE	Compilation Date	
EGSE_INTEGER_DERIVED_VALUE	Expression Code	
EGSE_INTEGER_DERIVED_VALUE	Expression Source	
EGSE_INTEGER_MEASUREMENT	Analog Calibration Coefficients	(if Calib Curve Type.Curve Type = POLYNOM)
EGSE_INTEGER_MEASUREMENT	Analog Point Pairs	(if Calib Curve Type.Curve Type = POINT_PAIRS)
EGSE_INTEGER_MEASUREMENT	Calib Curve Type	
EGSE_INTEGER_MEASUREMENT	Integer Raw Value Range	(if Raw Value Type.Raw Value Type = SIGNED_INTEGER)
EGSE_INTEGER_MEASUREMENT	Raw Value Size in Bits	(if Raw Value Type.Raw Value Type in (SIGNED_INTEGER, UNSIGNED_INTEGER))
EGSE_INTEGER_MEASUREMENT	Raw Value Type	

```
EGSE_INTEGER_MEASUREMENT         Unsigned Integer Raw Value Range    (if Raw Value Type.Raw Value Type
                                                                         = UNSIGNED_INTEGER)
EGSE_MONITOR_LIST                Value End Item List
EGSE_NODE                        CGS Internal Name                   (if Node Type.Node Type
                                                                         in (DATABASE_SERVER,
                                                                             SIMULATOR,
                                                                             TEST_NODE,
                                                                             WORKSTATION))
EGSE_NODE                        Logical Name                        (if Node Type.Node Type
                                                                         in (FRONT_END_EQUIPMENT,
                                                                             UNIT_UNDER_TEST))
EGSE_NODE                        Node Type
EGSE_PREDEFINED_TC               CCSDS Primary Header
EGSE_PREDEFINED_TC               EGSE Stimulus General Info
EGSE_SOFTWARE                    EGSE Software
EGSE_SOFTWARE                    Short Name                          (if EGSE Software.Software Type
                                                                         = SAS)
EGSE_STRING_DERIVED_VALUE        Compilation Date
EGSE_STRING_DERIVED_VALUE        Expression Code
EGSE_STRING_DERIVED_VALUE        Expression Source
EGSE_TEST_CONFIGURATION          EGSE Database Node
EGSE_USER_MESSAGE                Message Text
FWDU_COMPOSITE_BINARY            Raw data
FWDU_CONVERSION_TEXT             Text
FWDU_DATA_DEF_RECORD_TEXT        Text
FWDU_DYNAMIC_OBJECT_TABLE_TEXT   Text
FWDU_EQUIPMENT_CONSTRAINTS       Text
FWDU_GIF_BINARY                  Raw data
FWDU_HELP_TEXT                   Text
FWDU_LIBRARY_BINARY              Raw data
FWDU_LIST_TEXT                   Text
FWDU_SYMBOL_BITMAP_BINARY        Raw data
FWDU_SYMBOL_TABLE_TEXT           Text
FWDU_SYNOPTIC_DISPLAY            Compilation Date
FWDU_XWD_BINARY                  Raw data
FWDU_X_BITMAP_BINARY             Raw data
GDU_DESCRIPTION_LIST             Formal Parameter
GDU_DESCRIPTION_LIST             Stimuli List
HLCL_COMMAND_SEQUENCE            Compilation Date
INTEGER_CONSTANT                 General
INTEGER_STIMULUS                 Analog Decalibration Coefficients   (if Calib Curve Type.Curve Type
                                                                         = POLYNOM)
INTEGER_STIMULUS                 Analog Decalibration Point Pairs    (if Calib Curve Type.Curve Type
                                                                         = POINT_PAIRS)
INTEGER_STIMULUS                 Calib Curve Type
INTEGER_STIMULUS                 Formal Parameter
INTEGER_STIMULUS                 Integer Engineering Range
INTEGER_STIMULUS                 Integer Raw Value Range             (if Raw Value Type.Raw Value Type
                                                                         = SIGNED_INTEGER)
INTEGER_STIMULUS                 Raw Value Size in Bits              (if Raw Value Type.Raw Value Type
                                                                         in (SIGNED_INTEGER,
                                                                             UNSIGNED_INTEGER))
INTEGER_STIMULUS                 Raw Value Type
INTEGER_STIMULUS                 Unsigned Integer Raw Value Range    (if Raw Value Type.Raw Value Type
                                                                         = UNSIGNED_INTEGER)
MX_MODEL                         Compilation Date
PULSE_STIMULUS                   Boolean Calibration
PULSE_STIMULUS                   Integer Raw Value Range             (if Raw Value Type.Raw Value Type
                                                                         = SIGNED_INTEGER)
PULSE_STIMULUS                   Raw Value Size in Bits              (if Raw Value Type.Raw Value Type
                                                                         = SIGNED_INTEGER)
PULSE_STIMULUS                   Raw Value Type
REAL_CONSTANT                    General
RESPONSE_PACKET                  Response Packet
SIMULATED_ADU_DESCRIPTION        Referenced ADU
STRING_CONSTANT                  General
STRUCTURED_ADU_DESCRIPTION       ADU General Info
STRUCTURED_ADU_DESCRIPTION       Measurement End Items
SWEU                             Compilation Date
SWRU                             Compilation Date
UCL_AUTOMATED_PROCEDURE          Compilation Date
UCL_SYSTEM_LIBRARY               Compilation Date
```

```
UCL_USER_LIBRARY                 Compilation Date
UNSIGNED_INTEGER_MEASUREMENT     Analog Calibration Coefficients     (if Calib Curve Type.Curve Type
                                                                         = POLYNOM)
UNSIGNED_INTEGER_MEASUREMENT     Analog Point Pairs                  (if Calib Curve Type.Curve Type
                                                                         = POINT_PAIRS)
UNSIGNED_INTEGER_MEASUREMENT     Calib Curve Type
UNSIGNED_INTEGER_MEASUREMENT     Raw Value Size in Bits              (if Raw Value Type.Raw Value Type
                                                                         = UNSIGNED_INTEGER)
UNSIGNED_INTEGER_MEASUREMENT     Raw Value Type
UNSIGNED_INTEGER_MEASUREMENT     Unsigned Integer Engineering Range
UNSIGNED_INTEGER_MEASUREMENT     Unsigned Integer Raw Value Range    (if Raw Value Type.Raw Value Type
                                                                         = UNSIGNED_INTEGER)
UNSIGNED_INTEGER_STIMULUS        Analog Decalibration Coefficients   (if Calib Curve Type.Curve Type
                                                                         = POLYNOM)
UNSIGNED_INTEGER_STIMULUS        Analog Decalibration Point Pairs    (if Calib Curve Type.Curve Type
                                                                         = POINT_PAIRS)
UNSIGNED_INTEGER_STIMULUS        Calib Curve Type
UNSIGNED_INTEGER_STIMULUS        Formal Parameter
UNSIGNED_INTEGER_STIMULUS        Raw Value Size in Bits              (if Raw Value Type.Raw Value Type
                                                                         = UNSIGNED_INTEGER)
UNSIGNED_INTEGER_STIMULUS        Raw Value Type
UNSIGNED_INTEGER_STIMULUS        Unsigned Integer Engineering Range
UNSIGNED_INTEGER_STIMULUS        Unsigned Integer Raw Value Range    (if Raw Value Type.Raw Value Type
                                                                         = UNSIGNED_INTEGER)
UNSIGNED_INTEGER_SW_VARIABLE     Initial Unsigned Integer Value
UNSIGNED_INTEGER_SW_VARIABLE     Unsigned Integer Engineering Range
UNSTRUCTURED_ADU_DESCRIPTION     ADU General Info
UNSTRUCTURED_ADU_DESCRIPTION     Global Length
UNSTRUCTURED_ADU_DESCRIPTION     Measurement End Items
WDU_GROUND_SYNOPTIC_DISPLAY      Compilation Date
```

CC–2 Mandatory attribute is not defined

The following attributes are mandatory:

```
aggregate name                            attribute name
----------------------------------------  ----------------------------------------
ADU General Info                          Acquisition Rate
ADU General Info                          All Measurements with Physical Address
ADU General Info                          Global Physical Address Required
ADU General Info                          SAS Reference
APID Table                                Application ID (APID)
APID Table                                Destination CCSDS End Point
APID Table                                Source CCSDS End point
APID Table                                Type
Action Reference List                     CDU Internal Version
Analog Calibration Coefficients           Parameter Name
Analog Decalibration Point Pairs          Engineering Value
Analog Decalibration Point Pairs          Engineering Value
Analog Decalibration Point Pairs          Parameter Name
Analog Decalibration Point Pairs          Raw Value
Analog Decalibration Point Pairs          Raw Value
Analog Point Pairs                        Engineering Value
Analog Point Pairs                        Engineering Value
Analog Point Pairs                        Parameter Name
Analog Point Pairs                        Raw Value
Analog Point Pairs                        Raw Value
Binary Definition                         Position
Binary Definition                         Value
Binary Definition                         Value Type
Bitset Default Value                      Name
Bitset Default Value                      Value
Body Reference List                       CDU Internal Version
Body Reference List                       Pathname
Body Reference List                       Pathname SID
Boolean Calibration                       False
Boolean Calibration                       True
Boolean Constraints                       First
Boolean Constraints                       Last
Boolean Constraints                       Name
Boolean Default Value                     Name
```

Boolean Default Value	Value
Boolean Monitoring	Expected Value
Burst Pulse Default Value	Name
Burst Pulse Default Value	Value
Byte Stream Conditions	Action Enditem Reference
Byte Stream Conditions	Action Type
Byte Stream Conditions	Operator
Byte Stream Conditions	Value
Bytestream Calibration	Length
Bytestream Calibration	Position
CCSDS Primary Header	Application ID
CCSDS Primary Header	Packet Length
CCSDS Primary Header	Packet Type
CCSDS Primary Header	Secondary Header
CCSDS Primary Header	Sequence Flags
CCSDS Primary Header	Version Number
CCSDS Second Header	Checksum Indicator
CCSDS Second Header	Packet Type
CCSDS Second Header	Time ID
CGS Internal Name	CGS Internal Number
CGS Internal Name	CGS Prefix
Calib Curve Type	Curve Type
Character Constraints	First
Character Constraints	Last
Character Constraints	Name
Character Default Value	Name
Character Default Value	Value
Code	Code
Command Verification	Measurement to be checked
Command Verification	Operator
Command Verification	Value (Low Value for Range)
Compilation Date	Compilation Date
Compilation Date	Compilation Status
Completion Code Constraints	First
Completion Code Constraints	Last
Completion Code Constraints	Name
Completion Code Default Value	Name
Completion Code Default Value	Value
Crew Procedure Name	Group Name
Crew Procedure Name	Script Name
Crew Procedure Name	Subgroup Name
Cross Reference List	Pathname
Cross Reference List	Referenced CDU internal version
Cross Reference List	SID
Decalibration Curve Type	Curve Type
Decalibration Curve Type	Parameter Name
Discrete Calibration	Discrete Calibration Raw Value
Discrete Calibration	Discrete Calibration State Code
Discrete Conditions	Action Enditem Reference
Discrete Conditions	Action Type
Discrete Conditions	Operator
Discrete Conditions	Value
Discrete Decalibration	Parameter Name
Discrete Decalibration	Raw Value
Discrete Decalibration	State Code
Double Float Danger Limit	High Limit
Double Float Danger Limit	Low Limit
Double Float Engineering Range	High Value
Double Float Engineering Range	Low Value
Double Float Nominal Limits	High Value
Double Float Nominal Limits	Low Limit
Duration Constraints	First
Duration Constraints	Last
Duration Constraints	Name
Duration Default Value	Name
Duration Default Value	Value
EGSE Database Node	Database Node
EGSE Software	Software Type
EGSE Stimulus General Info	SAS Reference
Engineering Unit	Engineering Unit
Engineering Unit	Parameter Name
Engineering Units	Engineering Units

Expected Value	Expected Value
Expression Code	Expression Code
Expression Source	Expression Source
Float Conditions	Action Enditem Reference
Float Conditions	Action Type
Float Conditions	Operator
Float Conditions	Value 1 (Low Value)
Float Danger Limits	High Limit
Float Danger Limits	Low Limit
Float Definition	Float Value
Float Definition	Position
Float Engineering Range	High Value
Float Engineering Range	High Value
Float Engineering Range	Low Value
Float Engineering Range	Low Value
Float Engineering Range	Parameter Name
Float Nominal Limits	High Limit
Float Nominal Limits	Low Limit
Float Raw Value Range	High Value
Float Raw Value Range	High Value
Float Raw Value Range	Low Value
Float Raw Value Range	Low Value
Float Raw Value Range	Parameter Name
Formal Parameter	Dimension
Formal Parameter	List
Formal Parameter	Mode Out
Formal Parameter	Mode in
Formal Parameter	Name
Formal Parameter	Optional
Formal Parameter	SW Type
General	Exchange AP
General	Float Value
General	Initialization AP
General	Integer Value
General	String Value
General Bitstream Layout	Format
General Bitstream Layout	Location
General Bitstream Layout	Position
Global Length	Global Length
Housekeeping Source	Housekeeping Identifier
Initial Boolean Value	Initial Boolean Value
Initial Discrete Value	Initial Discrete Value
Initial Double Float Value	Initial Double Float Value
Initial Float Value	Initial Float Value
Initial Integer Value	Initial integer value
Initial Unsigned Integer Value	Initial unsigned integer value
Integer Conditions	Action Enditem Reference
Integer Conditions	Action Type
Integer Conditions	Operator
Integer Conditions	Value 1 (Low Value)
Integer Constraint	First
Integer Constraint	Last
Integer Constraint	Name
Integer Danger Limits	High Limit
Integer Danger Limits	Low Limit
Integer Default Value	Name
Integer Default Value	Value
Integer Definition	Integer Value
Integer Definition	Number of Bits
Integer Definition	Position
Integer Engineering Range	High Value
Integer Engineering Range	High Value
Integer Engineering Range	Low Value
Integer Engineering Range	Low Value
Integer Engineering Range	Parameter Name
Integer Nominal Limits	High Limit
Integer Nominal Limits	Low Limit
Integer Raw Value Range	High Value
Integer Raw Value Range	High Value
Integer Raw Value Range	Low Value
Integer Raw Value Range	Low Value
Integer Raw Value Range	Parameter Name

List of Parameters	Location
List of Parameters	Number of Bits
List of Parameters	Parameter Name
Logical Name	Logical Name
Long Real Constraints	First
Long Real Constraints	Last
Long Real Constraints	Name
Long Real Default Value	Name
Long Real Default Value	Value
Measurement End Items	End Item Reference
Measurement End Items	End Item Reference
Measurement End Items	Location
Object Code	Object Code
Parameterized Pathname	Name
Pathname Constraints	Item Type
Pathname Constraints	Name
Pathname Constraints	Number
Pathname Default Value	Name
Pathname Default Value	Value
Pulse Default Value	Name
Pulse Default Value	Value
Raw Value Size in Bits	Parameter Name
Raw Value Size in Bits	Raw Value Size in Bits
Raw Value Size in Bits	Raw Value Size in Bits
Raw Value Size in Bytes	Parameter Name
Raw Value Size in Bytes	Raw Value Size in Bytes
Raw Value Size in Bytes	Raw Value Size in Bytes
Raw Value Type	Parameter Name
Raw Value Type	Raw Value Type
Raw Value Type	Raw Value Type
Real Constraints	First
Real Constraints	Last
Real Constraints	Name
Real Default Value	Name
Real Default Value	Value
Reference List	Internal CDU version
Reference List	Pathname
Reference List	Pathname SID
Referenced ADU	ADU Reference
Response Packet	Response Packet Reference
SID Location List	SID Location List
SWRU Reference	SWRU Reference
Short Name	Short Name
Source	Source
State Code Constraints	Name
State Code Constraints	State Code Number
State Code Constraints	State Code Text
State Code Default Value	Name
State Code Default Value	Value
State Code List	Index
Stimuli List	End Item Reference
String Constraints	Name
String Constraints	String Size
String Default Value	Name
String Default Value	Value
Subitem Pathname Constraints	Name
Subitem Pathname Constraints	Number
Subitem Pathname Constraints	Subitem Type
Subitem Pathname Default Value	Item Value
Subitem Pathname Default Value	Name
Subitem Pathname Default Value	Subitem Value
Symbol Reference List	CDU Internal Version
TC End Item References	Location
TC End Item References	Stimulus Reference
Time Constraints	First Day
Time Constraints	First Month
Time Constraints	First Second
Time Constraints	First Year
Time Constraints	Last Day
Time Constraints	Last Month
Time Constraints	Last Second
Time Constraints	Last Year

Time Constraints	Name
Time Default Value	Day
Time Default Value	Month
Time Default Value	Name
Time Default Value	Second
Time Default Value	Year
Unsigned Integer Constraints	First
Unsigned Integer Constraints	Last
Unsigned Integer Constraints	Name
Unsigned Integer Danger Limits	High Limit
Unsigned Integer Danger Limits	Low Limit
Unsigned Integer Default Value	Name
Unsigned Integer Default Value	Value
Unsigned Integer Engineering Range	High Value
Unsigned Integer Engineering Range	Low Value
Unsigned Integer Nominal Limits	High Limit
Unsigned Integer Nominal Limits	Low Limit
Unsigned Integer Raw Value Range	High Value
Unsigned Integer Raw Value Range	High Value
Unsigned Integer Raw Value Range	Low Value
Unsigned Integer Raw Value Range	Low Value
Unsigned Integer Raw Value Range	Parameter Name
Value End Item List	End Item Reference
Variable Reference List	CDU Internal Version
Variable Reference List	EGSE Pathname
Variable Reference List	EGSE SID
Word Code Default Value	Name
Word Code Default Value	Value

N-1.2 Uniqueness Checks

CC–3 Attribute is not unique in CCU
Attribute is not unique in CDU

The following attributes must be unique in the configuration scope:

aggregate name	attribute name
Short Name	Short Name

CC–4 Attribute is not unique in repeating agg

The following attributes must be unique in the end item:

aggregate name	attribute name
Analog Calibration Coefficients	Parameter Name
Decalibration Curve Type	Parameter Name
Discrete Calibration	Discrete Calibration Raw Value
Discrete Calibration	Discrete Calibration State Code
EGSE Test Nodes	Test Node
EGSE Workstation Nodes	Workstation Node
Engineering Unit	Parameter Name
Float Engineering Range	Parameter Name
Float Raw Value Range	Parameter Name
Formal Parameter	Name
Integer Engineering Range	Parameter Name
Integer Raw Value Range	Parameter Name
List of Parameters	Parameter Name
Raw Value Size in Bits	Parameter Name
Raw Value Size in Bytes	Parameter Name
Raw Value Type	Parameter Name
Unsigned Integer Raw Value Range	Parameter Name

CC–5 Foreign key aggr. is not unique in CCU
Foreign key aggr. is not unique in CDU

The following foreign key aggregates must be unique in the configuration scope:

Aggregate name
Foreign Key

N-1.3 Referential Integrity Checks

CC–6 Pathname reference out of scope

- Constraint: Each pathname must refer to an item within the configuration scope.

CC–7 Pathname reference to wrong type in CCU
Pathname reference to wrong type in CDU

The following attributes of type pathname must refer to an item of correct type:

aggregate name	attribute name	type name
ADU General Info	SAS Reference	EGSE_SOFTWARE
APID Table	Destination CCSDS End Point	CCSDS_END_POINT
APID Table	Source CCSDS End point	CCSDS_END_POINT
Boolean Monitoring	Exception Action	EGSE_ANALOG_STIMULUS
Boolean Monitoring	Exception Action	EGSE_BINARY_PACKET
Boolean Monitoring	Exception Action	EGSE_DISCRETE_STIMULUS
Boolean Monitoring	Exception Action	EGSE_PREDEFINED_TC
Boolean Monitoring	Exception Action	GDU_DESCRIPTION_LIST
Boolean Monitoring	Exception Action	UCL_AUTOMATED_PROCEDURE
Boolean Monitoring	Exception Message	EGSE_USER_MESSAGE
Byte Stream Conditions	Action Enditem Reference	CCSDS_ADU_DESCRIPTION
Byte Stream Conditions	Action Enditem Reference	CDU
Byte Stream Conditions	Action Enditem Reference	EGSE_BYTE_STREAM_MEASUREMENT
Byte Stream Conditions	Action Enditem Reference	EGSE_BYTE_STREAM_SW_VARIABLE
Byte Stream Conditions	Action Enditem Reference	EGSE_DISCRETE_DERIVED_VALUE
Byte Stream Conditions	Action Enditem Reference	EGSE_DISCRETE_MEASUREMENT
Byte Stream Conditions	Action Enditem Reference	EGSE_DISCRETE_SW_VARIABLE
Byte Stream Conditions	Action Enditem Reference	EGSE_FLOAT_DERIVED_VALUE
Byte Stream Conditions	Action Enditem Reference	EGSE_FLOAT_MEASUREMENT
Byte Stream Conditions	Action Enditem Reference	EGSE_FLOAT_SW_VARIABLE
Byte Stream Conditions	Action Enditem Reference	EGSE_INTEGER_DERIVED_VALUE
Byte Stream Conditions	Action Enditem Reference	EGSE_INTEGER_MEASUREMENT
Byte Stream Conditions	Action Enditem Reference	EGSE_INTEGER_SW_VARIABLE
Byte Stream Conditions	Action Enditem Reference	EGSE_MONITOR_LIST
Byte Stream Conditions	Action Enditem Reference	EGSE_STRING_DERIVED_VALUE
Byte Stream Conditions	Action Enditem Reference	STRUCTURED_ADU_DESCRIPTION
Byte Stream Conditions	Action Enditem Reference	UCL_AUTOMATED_PROCEDURE
Byte Stream Conditions	Action Enditem Reference	UNSTRUCTURED_ADU_DESCRIPTION
Byte Stream Conditions	Action Enditem Reference	VIRTUAL
Byte Stream Monitor List	Exception Action	EGSE_ANALOG_STIMULUS
Byte Stream Monitor List	Exception Action	EGSE_BINARY_PACKET
Byte Stream Monitor List	Exception Action	EGSE_DISCRETE_STIMULUS
Byte Stream Monitor List	Exception Action	EGSE_PREDEFINED_TC
Byte Stream Monitor List	Exception Action	GDU_DESCRIPTION_LIST
Byte Stream Monitor List	Exception Action	UCL_AUTOMATED_PROCEDURE
Byte Stream Monitor List	Exception Message	EGSE_USER_MESSAGE
Command Verification	Measurement to be checked	EGSE_BYTE_STREAM_MEASUREMENT
Command Verification	Measurement to be checked	EGSE_BYTE_STREAM_SW_VARIABLE
Command Verification	Measurement to be checked	EGSE_DISCRETE_DERIVED_VALUE
Command Verification	Measurement to be checked	EGSE_DISCRETE_MEASUREMENT
Command Verification	Measurement to be checked	EGSE_DISCRETE_SW_VARIABLE
Command Verification	Measurement to be checked	EGSE_FLOAT_DERIVED_VALUE
Command Verification	Measurement to be checked	EGSE_FLOAT_MEASUREMENT
Command Verification	Measurement to be checked	EGSE_FLOAT_SW_VARIABLE

Command Verification	Measurement to be checked	EGSE_INTEGER_DERIVED_VALUE
Command Verification	Measurement to be checked	EGSE_INTEGER_MEASUREMENT
Command Verification	Measurement to be checked	EGSE_INTEGER_SW_VARIABLE
Command Verification	Measurement to be checked	EGSE_STRING_DERIVED_VALUE
Demo References	Reference to a mapped Type	DEMO_DISCRETE_MEASUREMENT
Demo References	Reference to a unmapped Type	EGSE_DISCRETE_MEASUREMENT
Discrete Conditions	Action Enditem Reference	CCSDS_ADU_DESCRIPTION
Discrete Conditions	Action Enditem Reference	CDU
Discrete Conditions	Action Enditem Reference	EGSE_BYTE_STREAM_MEASUREMENT
Discrete Conditions	Action Enditem Reference	EGSE_BYTE_STREAM_SW_VARIABLE
Discrete Conditions	Action Enditem Reference	EGSE_DISCRETE_DERIVED_VALUE
Discrete Conditions	Action Enditem Reference	EGSE_DISCRETE_MEASUREMENT
Discrete Conditions	Action Enditem Reference	EGSE_DISCRETE_SW_VARIABLE
Discrete Conditions	Action Enditem Reference	EGSE_FLOAT_DERIVED_VALUE
Discrete Conditions	Action Enditem Reference	EGSE_FLOAT_MEASUREMENT
Discrete Conditions	Action Enditem Reference	EGSE_FLOAT_SW_VARIABLE
Discrete Conditions	Action Enditem Reference	EGSE_INTEGER_DERIVED_VALUE
Discrete Conditions	Action Enditem Reference	EGSE_INTEGER_MEASUREMENT
Discrete Conditions	Action Enditem Reference	EGSE_INTEGER_SW_VARIABLE
Discrete Conditions	Action Enditem Reference	EGSE_MONITOR_LIST
Discrete Conditions	Action Enditem Reference	EGSE_STRING_DERIVED_VALUE
Discrete Conditions	Action Enditem Reference	STRUCTURED_ADU_DESCRIPTION
Discrete Conditions	Action Enditem Reference	UCL_AUTOMATED_PROCEDURE
Discrete Conditions	Action Enditem Reference	UNSTRUCTURED_ADU_DESCRIPTION
Discrete Conditions	Action Enditem Reference	VIRTUAL
Double Float Danger Limit	Delta Action	EGSE_ANALOG_STIMULUS
Double Float Danger Limit	Delta Action	EGSE_BINARY_PACKET
Double Float Danger Limit	Delta Action	EGSE_DISCRETE_STIMULUS
Double Float Danger Limit	Delta Action	EGSE_PREDEFINED_TC
Double Float Danger Limit	Delta Action	GDU_DESCRIPTION_LIST
Double Float Danger Limit	Delta Action	UCL_AUTOMATED_PROCEDURE
Double Float Danger Limit	Delta Message	EGSE_USER_MESSAGE
Double Float Danger Limit	High Action	EGSE_ANALOG_STIMULUS
Double Float Danger Limit	High Action	EGSE_BINARY_PACKET
Double Float Danger Limit	High Action	EGSE_DISCRETE_STIMULUS
Double Float Danger Limit	High Action	EGSE_PREDEFINED_TC
Double Float Danger Limit	High Action	GDU_DESCRIPTION_LIST
Double Float Danger Limit	High Action	UCL_AUTOMATED_PROCEDURE
Double Float Danger Limit	High Message	EGSE_USER_MESSAGE
Double Float Danger Limit	Low Action	EGSE_ANALOG_STIMULUS
Double Float Danger Limit	Low Action	EGSE_BINARY_PACKET
Double Float Danger Limit	Low Action	EGSE_DISCRETE_STIMULUS
Double Float Danger Limit	Low Action	EGSE_PREDEFINED_TC
Double Float Danger Limit	Low Action	GDU_DESCRIPTION_LIST
Double Float Danger Limit	Low Action	UCL_AUTOMATED_PROCEDURE
Double Float Danger Limit	Low Message	EGSE_USER_MESSAGE
Double Float Nominal Limits	Delta Action	EGSE_ANALOG_STIMULUS
Double Float Nominal Limits	Delta Action	EGSE_BINARY_PACKET
Double Float Nominal Limits	Delta Action	EGSE_DISCRETE_STIMULUS
Double Float Nominal Limits	Delta Action	EGSE_PREDEFINED_TC
Double Float Nominal Limits	Delta Action	GDU_DESCRIPTION_LIST
Double Float Nominal Limits	Delta Action	UCL_AUTOMATED_PROCEDURE
Double Float Nominal Limits	Delta Message	EGSE_USER_MESSAGE
Double Float Nominal Limits	High Action	EGSE_ANALOG_STIMULUS
Double Float Nominal Limits	High Action	EGSE_BINARY_PACKET
Double Float Nominal Limits	High Action	EGSE_DISCRETE_STIMULUS
Double Float Nominal Limits	High Action	EGSE_PREDEFINED_TC
Double Float Nominal Limits	High Action	GDU_DESCRIPTION_LIST
Double Float Nominal Limits	High Action	UCL_AUTOMATED_PROCEDURE
Double Float Nominal Limits	High Message	EGSE_USER_MESSAGE
Double Float Nominal Limits	Low Action	EGSE_ANALOG_STIMULUS
Double Float Nominal Limits	Low Action	EGSE_BINARY_PACKET
Double Float Nominal Limits	Low Action	EGSE_DISCRETE_STIMULUS
Double Float Nominal Limits	Low Action	EGSE_PREDEFINED_TC
Double Float Nominal Limits	Low Action	GDU_DESCRIPTION_LIST
Double Float Nominal Limits	Low Action	UCL_AUTOMATED_PROCEDURE
Double Float Nominal Limits	Low Message	EGSE_USER_MESSAGE
EGSE Database Node	Database Node	EGSE_NODE
EGSE Software	SWEU–Reference	SWEU
EGSE Stimulus General Info	SAS Reference	EGSE_SOFTWARE
EGSE Test Node Items	Loaded Item	APID
EGSE Test Node Items	Loaded Item	CCSDS_ADU_DESCRIPTION

EGSE Test Node Items	Loaded Item	CDU
EGSE Test Node Items	Loaded Item	EGSE_ANALOG_STIMULUS
EGSE Test Node Items	Loaded Item	EGSE_BINARY_PACKET
EGSE Test Node Items	Loaded Item	EGSE_BYTE_STREAM_SW_VARIABLE
EGSE Test Node Items	Loaded Item	EGSE_DISCRETE_DERIVED_VALUE
EGSE Test Node Items	Loaded Item	EGSE_DISCRETE_STIMULUS
EGSE Test Node Items	Loaded Item	EGSE_DISCRETE_SW_VARIABLE
EGSE Test Node Items	Loaded Item	EGSE_FLOAT_DERIVED_VALUE
EGSE Test Node Items	Loaded Item	EGSE_FLOAT_SW_VARIABLE
EGSE Test Node Items	Loaded Item	EGSE_INTEGER_DERIVED_VALUE
EGSE Test Node Items	Loaded Item	EGSE_INTEGER_SW_VARIABLE
EGSE Test Node Items	Loaded Item	EGSE_MONITOR_LIST
EGSE Test Node Items	Loaded Item	EGSE_PREDEFINED_TC
EGSE Test Node Items	Loaded Item	EGSE_SOFTWARE
EGSE Test Node Items	Loaded Item	EGSE_STRING_DERIVED_VALUE
EGSE Test Node Items	Loaded Item	EGSE_USER_MESSAGE
EGSE Test Node Items	Loaded Item	GDU_DESCRIPTION_LIST
EGSE Test Node Items	Loaded Item	RESPONSE_PACKET
EGSE Test Node Items	Loaded Item	SIMULATED_ADU_DESCRIPTION
EGSE Test Node Items	Loaded Item	STRUCTURED_ADU_DESCRIPTION
EGSE Test Node Items	Loaded Item	SWOP_COMMAND
EGSE Test Node Items	Loaded Item	UCL_USER_LIBRARY
EGSE Test Node Items	Loaded Item	UNSTRUCTURED_ADU_DESCRIPTION
EGSE Test Node Items	Loaded Item	VIRTUAL
EGSE Test Node Items	Test Node	EGSE_NODE
EGSE Test Node SASs	Test Node	EGSE_NODE
EGSE Test Node SASs	Used SAS	EGSE_SOFTWARE
EGSE Test Nodes	Initial Automated Procedure	UCL_AUTOMATED_PROCEDURE
EGSE Test Nodes	Overview Synoptic	WDU_GROUND_SYNOPTIC_DISPLAY
EGSE Test Nodes	Test Node	EGSE_NODE
EGSE Workstation Nodes	Workstation Node	EGSE_NODE
Expected Value	Exception Action	EGSE_ANALOG_STIMULUS
Expected Value	Exception Action	EGSE_BINARY_PACKET
Expected Value	Exception Action	EGSE_DISCRETE_STIMULUS
Expected Value	Exception Action	EGSE_PREDEFINED_TC
Expected Value	Exception Action	GDU_DESCRIPTION_LIST
Expected Value	Exception Action	UCL_AUTOMATED_PROCEDURE
Expected Value	Exception Message	EGSE_USER_MESSAGE
Float Conditions	Action Enditem Reference	CCSDS_ADU_DESCRIPTION
Float Conditions	Action Enditem Reference	CDU
Float Conditions	Action Enditem Reference	EGSE_BYTE_STREAM_MEASUREMENT
Float Conditions	Action Enditem Reference	EGSE_BYTE_STREAM_SW_VARIABLE
Float Conditions	Action Enditem Reference	EGSE_DISCRETE_DERIVED_VALUE
Float Conditions	Action Enditem Reference	EGSE_DISCRETE_MEASUREMENT
Float Conditions	Action Enditem Reference	EGSE_DISCRETE_SW_VARIABLE
Float Conditions	Action Enditem Reference	EGSE_FLOAT_DERIVED_VALUE
Float Conditions	Action Enditem Reference	EGSE_FLOAT_MEASUREMENT
Float Conditions	Action Enditem Reference	EGSE_FLOAT_SW_VARIABLE
Float Conditions	Action Enditem Reference	EGSE_INTEGER_DERIVED_VALUE
Float Conditions	Action Enditem Reference	EGSE_INTEGER_MEASUREMENT
Float Conditions	Action Enditem Reference	EGSE_INTEGER_SW_VARIABLE
Float Conditions	Action Enditem Reference	EGSE_MONITOR_LIST
Float Conditions	Action Enditem Reference	EGSE_STRING_DERIVED_VALUE
Float Conditions	Action Enditem Reference	STRUCTURED_ADU_DESCRIPTION
Float Conditions	Action Enditem Reference	UCL_AUTOMATED_PROCEDURE
Float Conditions	Action Enditem Reference	UNSTRUCTURED_ADU_DESCRIPTION
Float Conditions	Action Enditem Reference	VIRTUAL
Float Danger Limits	Delta Action	EGSE_ANALOG_STIMULUS
Float Danger Limits	Delta Action	EGSE_BINARY_PACKET
Float Danger Limits	Delta Action	EGSE_DISCRETE_STIMULUS
Float Danger Limits	Delta Action	EGSE_PREDEFINED_TC
Float Danger Limits	Delta Action	GDU_DESCRIPTION_LIST
Float Danger Limits	Delta Action	UCL_AUTOMATED_PROCEDURE
Float Danger Limits	Delta Message	EGSE_USER_MESSAGE
Float Danger Limits	High Action	EGSE_ANALOG_STIMULUS
Float Danger Limits	High Action	EGSE_BINARY_PACKET
Float Danger Limits	High Action	EGSE_DISCRETE_STIMULUS
Float Danger Limits	High Action	EGSE_PREDEFINED_TC
Float Danger Limits	High Action	GDU_DESCRIPTION_LIST
Float Danger Limits	High Action	UCL_AUTOMATED_PROCEDURE
Float Danger Limits	High Message	EGSE_USER_MESSAGE
Float Danger Limits	Low Action	EGSE_ANALOG_STIMULUS

Float Danger Limits	Low Action	EGSE_BINARY_PACKET
Float Danger Limits	Low Action	EGSE_DISCRETE_STIMULUS
Float Danger Limits	Low Action	EGSE_PREDEFINED_TC
Float Danger Limits	Low Action	GDU_DESCRIPTION_LIST
Float Danger Limits	Low Action	UCL_AUTOMATED_PROCEDURE
Float Danger Limits	Low Message	EGSE_USER_MESSAGE
Float Nominal Limits	Delta Action	EGSE_ANALOG_STIMULUS
Float Nominal Limits	Delta Action	EGSE_BINARY_PACKET
Float Nominal Limits	Delta Action	EGSE_DISCRETE_STIMULUS
Float Nominal Limits	Delta Action	EGSE_PREDEFINED_TC
Float Nominal Limits	Delta Action	GDU_DESCRIPTION_LIST
Float Nominal Limits	Delta Action	UCL_AUTOMATED_PROCEDURE
Float Nominal Limits	Delta Message	EGSE_USER_MESSAGE
Float Nominal Limits	High Action	EGSE_ANALOG_STIMULUS
Float Nominal Limits	High Action	EGSE_BINARY_PACKET
Float Nominal Limits	High Action	EGSE_DISCRETE_STIMULUS
Float Nominal Limits	High Action	EGSE_PREDEFINED_TC
Float Nominal Limits	High Action	GDU_DESCRIPTION_LIST
Float Nominal Limits	High Action	UCL_AUTOMATED_PROCEDURE
Float Nominal Limits	High Message	EGSE_USER_MESSAGE
Float Nominal Limits	Low Action	EGSE_ANALOG_STIMULUS
Float Nominal Limits	Low Action	EGSE_BINARY_PACKET
Float Nominal Limits	Low Action	EGSE_DISCRETE_STIMULUS
Float Nominal Limits	Low Action	EGSE_PREDEFINED_TC
Float Nominal Limits	Low Action	GDU_DESCRIPTION_LIST
Float Nominal Limits	Low Action	UCL_AUTOMATED_PROCEDURE
Float Nominal Limits	Low Message	EGSE_USER_MESSAGE
General	Exchange AP	UCL_AUTOMATED_PROCEDURE
General	Initialization AP	UCL_AUTOMATED_PROCEDURE
General	Reference	CDU
General	Reference	VIRTUAL
Integer Conditions	Action Enditem Reference	CCSDS_ADU_DESCRIPTION
Integer Conditions	Action Enditem Reference	CDU
Integer Conditions	Action Enditem Reference	EGSE_BYTE_STREAM_MEASUREMENT
Integer Conditions	Action Enditem Reference	EGSE_BYTE_STREAM_SW_VARIABLE
Integer Conditions	Action Enditem Reference	EGSE_DISCRETE_DERIVED_VALUE
Integer Conditions	Action Enditem Reference	EGSE_DISCRETE_MEASUREMENT
Integer Conditions	Action Enditem Reference	EGSE_DISCRETE_SW_VARIABLE
Integer Conditions	Action Enditem Reference	EGSE_FLOAT_DERIVED_VALUE
Integer Conditions	Action Enditem Reference	EGSE_FLOAT_MEASUREMENT
Integer Conditions	Action Enditem Reference	EGSE_FLOAT_SW_VARIABLE
Integer Conditions	Action Enditem Reference	EGSE_INTEGER_DERIVED_VALUE
Integer Conditions	Action Enditem Reference	EGSE_INTEGER_MEASUREMENT
Integer Conditions	Action Enditem Reference	EGSE_INTEGER_SW_VARIABLE
Integer Conditions	Action Enditem Reference	EGSE_MONITOR_LIST
Integer Conditions	Action Enditem Reference	EGSE_STRING_DERIVED_VALUE
Integer Conditions	Action Enditem Reference	STRUCTURED_ADU_DESCRIPTION
Integer Conditions	Action Enditem Reference	UCL_AUTOMATED_PROCEDURE
Integer Conditions	Action Enditem Reference	UNSTRUCTURED_ADU_DESCRIPTION
Integer Conditions	Action Enditem Reference	VIRTUAL
Integer Danger Limits	Delta Action	EGSE_ANALOG_STIMULUS
Integer Danger Limits	Delta Action	EGSE_BINARY_PACKET
Integer Danger Limits	Delta Action	EGSE_DISCRETE_STIMULUS
Integer Danger Limits	Delta Action	EGSE_PREDEFINED_TC
Integer Danger Limits	Delta Action	GDU_DESCRIPTION_LIST
Integer Danger Limits	Delta Action	UCL_AUTOMATED_PROCEDURE
Integer Danger Limits	Delta Message	EGSE_USER_MESSAGE
Integer Danger Limits	High Action	EGSE_ANALOG_STIMULUS
Integer Danger Limits	High Action	EGSE_BINARY_PACKET
Integer Danger Limits	High Action	EGSE_DISCRETE_STIMULUS
Integer Danger Limits	High Action	EGSE_PREDEFINED_TC
Integer Danger Limits	High Action	GDU_DESCRIPTION_LIST
Integer Danger Limits	High Action	UCL_AUTOMATED_PROCEDURE
Integer Danger Limits	High Message	EGSE_USER_MESSAGE
Integer Danger Limits	Low Action	EGSE_ANALOG_STIMULUS
Integer Danger Limits	Low Action	EGSE_BINARY_PACKET
Integer Danger Limits	Low Action	EGSE_DISCRETE_STIMULUS
Integer Danger Limits	Low Action	EGSE_PREDEFINED_TC
Integer Danger Limits	Low Action	GDU_DESCRIPTION_LIST
Integer Danger Limits	Low Action	UCL_AUTOMATED_PROCEDURE
Integer Danger Limits	Low Message	EGSE_USER_MESSAGE
Integer Nominal Limits	Delta Action	EGSE_ANALOG_STIMULUS

Integer Nominal Limits	Delta Action	EGSE_BINARY_PACKET
Integer Nominal Limits	Delta Action	EGSE_DISCRETE_STIMULUS
Integer Nominal Limits	Delta Action	EGSE_PREDEFINED_TC
Integer Nominal Limits	Delta Action	GDU_DESCRIPTION_LIST
Integer Nominal Limits	Delta Action	UCL_AUTOMATED_PROCEDURE
Integer Nominal Limits	Delta Message	EGSE_USER_MESSAGE
Integer Nominal Limits	High Action	EGSE_ANALOG_STIMULUS
Integer Nominal Limits	High Action	EGSE_BINARY_PACKET
Integer Nominal Limits	High Action	EGSE_DISCRETE_STIMULUS
Integer Nominal Limits	High Action	EGSE_PREDEFINED_TC
Integer Nominal Limits	High Action	GDU_DESCRIPTION_LIST
Integer Nominal Limits	High Action	UCL_AUTOMATED_PROCEDURE
Integer Nominal Limits	High Message	EGSE_USER_MESSAGE
Integer Nominal Limits	Low Action	EGSE_ANALOG_STIMULUS
Integer Nominal Limits	Low Action	EGSE_BINARY_PACKET
Integer Nominal Limits	Low Action	EGSE_DISCRETE_STIMULUS
Integer Nominal Limits	Low Action	EGSE_PREDEFINED_TC
Integer Nominal Limits	Low Action	GDU_DESCRIPTION_LIST
Integer Nominal Limits	Low Action	UCL_AUTOMATED_PROCEDURE
Integer Nominal Limits	Low Message	EGSE_USER_MESSAGE
Measurement End Items	End Item Reference	EGSE_BYTE_STREAM_MEASUREMENT
Measurement End Items	End Item Reference	EGSE_BYTE_STREAM_MEASUREMENT
Measurement End Items	End Item Reference	EGSE_DISCRETE_MEASUREMENT
Measurement End Items	End Item Reference	EGSE_DISCRETE_MEASUREMENT
Measurement End Items	End Item Reference	EGSE_FLOAT_MEASUREMENT
Measurement End Items	End Item Reference	EGSE_FLOAT_MEASUREMENT
Measurement End Items	End Item Reference	EGSE_INTEGER_MEASUREMENT
Measurement End Items	End Item Reference	EGSE_INTEGER_MEASUREMENT
Pathname Default Value	Value	BOOLEAN_MEASUREMENT
Pathname Default Value	Value	BOOLEAN_STIMULUS
Pathname Default Value	Value	BOOLEAN_SW_VARIABLE
Pathname Default Value	Value	EGSE_ANALOG_STIMULUS
Pathname Default Value	Value	EGSE_DISCRETE_MEASUREMENT
Pathname Default Value	Value	EGSE_DISCRETE_STIMULUS
Pathname Default Value	Value	EGSE_DISCRETE_SW_VARIABLE
Pathname Default Value	Value	EGSE_FLOAT_MEASUREMENT
Pathname Default Value	Value	EGSE_FLOAT_SW_VARIABLE
Pathname Default Value	Value	EGSE_INTEGER_SW_VARIABLE
Pathname Default Value	Value	STRING_CONSTANT
Referenced ADU	ADU Reference	CCSDS_ADU_DESCRIPTION
Referenced ADU	ADU Reference	STRUCTURED_ADU_DESCRIPTION
Referenced ADU	ADU Reference	UNSTRUCTURED_ADU_DESCRIPTION
Response Packet	Response Packet Reference	RESPONSE_PACKET
Response Packet	SAS Reference	EGSE_SOFTWARE
SWRU Reference	SWRU Reference	SWRU
Simulator Nodes	Simulator Node	EGSE_NODE
Stimuli List	End Item Reference	EGSE_ANALOG_STIMULUS
Stimuli List	End Item Reference	EGSE_DISCRETE_STIMULUS
Stimuli List	End Item Reference	EGSE_PREDEFINED_TC
TC End Item References	Stimulus Reference	BOOLEAN_STIMULUS
TC End Item References	Stimulus Reference	BURST_PULSE_STIMULUS
TC End Item References	Stimulus Reference	DOUBLE_FLOAT_STIMULUS
TC End Item References	Stimulus Reference	EGSE_ANALOG_STIMULUS
TC End Item References	Stimulus Reference	EGSE_DISCRETE_STIMULUS
TC End Item References	Stimulus Reference	INTEGER_STIMULUS
TC End Item References	Stimulus Reference	PULSE_STIMULUS
TC End Item References	Stimulus Reference	UNSIGNED_INTEGER_STIMULUS
Unsigned Integer Danger Limits	Delta Action	EGSE_ANALOG_STIMULUS
Unsigned Integer Danger Limits	Delta Action	EGSE_BINARY_PACKET
Unsigned Integer Danger Limits	Delta Action	EGSE_DISCRETE_STIMULUS
Unsigned Integer Danger Limits	Delta Action	EGSE_PREDEFINED_TC
Unsigned Integer Danger Limits	Delta Action	GDU_DESCRIPTION_LIST
Unsigned Integer Danger Limits	Delta Action	UCL_AUTOMATED_PROCEDURE
Unsigned Integer Danger Limits	Delta Message	EGSE_USER_MESSAGE
Unsigned Integer Danger Limits	High Action	EGSE_ANALOG_STIMULUS
Unsigned Integer Danger Limits	High Action	EGSE_BINARY_PACKET
Unsigned Integer Danger Limits	High Action	EGSE_DISCRETE_STIMULUS
Unsigned Integer Danger Limits	High Action	EGSE_PREDEFINED_TC
Unsigned Integer Danger Limits	High Action	GDU_DESCRIPTION_LIST
Unsigned Integer Danger Limits	High Action	UCL_AUTOMATED_PROCEDURE
Unsigned Integer Danger Limits	High Message	EGSE_USER_MESSAGE
Unsigned Integer Danger Limits	Low Action	EGSE_ANALOG_STIMULUS

Unsigned Integer Danger Limits	Low Action	EGSE_BINARY_PACKET
Unsigned Integer Danger Limits	Low Action	EGSE_DISCRETE_STIMULUS
Unsigned Integer Danger Limits	Low Action	EGSE_PREDEFINED_TC
Unsigned Integer Danger Limits	Low Action	GDU_DESCRIPTION_LIST
Unsigned Integer Danger Limits	Low Action	UCL_AUTOMATED_PROCEDURE
Unsigned Integer Danger Limits	Low Message	EGSE_USER_MESSAGE
Unsigned Integer Nominal Limits	Delta Action	EGSE_ANALOG_STIMULUS
Unsigned Integer Nominal Limits	Delta Action	EGSE_BINARY_PACKET
Unsigned Integer Nominal Limits	Delta Action	EGSE_DISCRETE_STIMULUS
Unsigned Integer Nominal Limits	Delta Action	EGSE_PREDEFINED_TC
Unsigned Integer Nominal Limits	Delta Action	GDU_DESCRIPTION_LIST
Unsigned Integer Nominal Limits	Delta Action	UCL_AUTOMATED_PROCEDURE
Unsigned Integer Nominal Limits	Delta Message	EGSE_USER_MESSAGE
Unsigned Integer Nominal Limits	High Action	EGSE_ANALOG_STIMULUS
Unsigned Integer Nominal Limits	High Action	EGSE_BINARY_PACKET
Unsigned Integer Nominal Limits	High Action	EGSE_DISCRETE_STIMULUS
Unsigned Integer Nominal Limits	High Action	EGSE_PREDEFINED_TC
Unsigned Integer Nominal Limits	High Action	GDU_DESCRIPTION_LIST
Unsigned Integer Nominal Limits	High Action	UCL_AUTOMATED_PROCEDURE
Unsigned Integer Nominal Limits	High Message	EGSE_USER_MESSAGE
Unsigned Integer Nominal Limits	Low Action	EGSE_ANALOG_STIMULUS
Unsigned Integer Nominal Limits	Low Action	EGSE_BINARY_PACKET
Unsigned Integer Nominal Limits	Low Action	EGSE_DISCRETE_STIMULUS
Unsigned Integer Nominal Limits	Low Action	EGSE_PREDEFINED_TC
Unsigned Integer Nominal Limits	Low Action	GDU_DESCRIPTION_LIST
Unsigned Integer Nominal Limits	Low Action	UCL_AUTOMATED_PROCEDURE
Unsigned Integer Nominal Limits	Low Message	EGSE_USER_MESSAGE
Value End Item List	End Item Reference	EGSE_BYTE_STREAM_MEASUREMENT
Value End Item List	End Item Reference	EGSE_BYTE_STREAM_SW_VARIABLE
Value End Item List	End Item Reference	EGSE_DISCRETE_DERIVED_VALUE
Value End Item List	End Item Reference	EGSE_DISCRETE_MEASUREMENT
Value End Item List	End Item Reference	EGSE_DISCRETE_SW_VARIABLE
Value End Item List	End Item Reference	EGSE_FLOAT_DERIVED_VALUE
Value End Item List	End Item Reference	EGSE_FLOAT_MEASUREMENT
Value End Item List	End Item Reference	EGSE_FLOAT_SW_VARIABLE
Value End Item List	End Item Reference	EGSE_INTEGER_DERIVED_VALUE
Value End Item List	End Item Reference	EGSE_INTEGER_MEASUREMENT
Value End Item List	End Item Reference	EGSE_INTEGER_SW_VARIABLE
Value End Item List	End Item Reference	EGSE_STRING_DERIVED_VALUE

N-1.4 Cross Reference Checks

The following checks are defined for end item types having Cross Reference Lists.

CC–8 Compilation status is FALSE – Regeneration of this item required

- Constraint: The compilation status must be true.

CC–9 Circular reference –> [pathname] –> ...

- Constraint: Circular references are not allowed for end items of the following types:
  UCL_AUTOMATED_PROCEDURE
  UCL_USER_LIBRARY

CC–10 Pathname reference [pathname] out of date – Regeneration of this item required

The following table describes the out of date rules of the consistency checker.
The table consists out of three colums:

1. End Item Type:	End item type that has a Cross Refrence List Aggregate, e.g. FWDU_SYNOPTIC_DISPLAY
2. Referenced End Item Type:	Referenced end item type, e.g. a FWDU_SYNOPTIC_DISPLAY may reference an end item of type EGSE_INTEGER_MEASUREMENT
3. Depending Aggregate of referenced end item type:	In case one of the attributes of this aggreagte changes, the end item has to be regenerated, e.g. a FWDU_SYNOPTIC_DISPLAY may reference an end item of type EGSE_INTEGER_MEASUREMENT. The EGSE_INTEGER_MEASUREMENT owns the aggerate T_INTEGER_DANGER_LIMITS. If one of the attributes of the aggreagte changes, the FWDU_SYNOPTIC_DISPLAY has to be regenerated.

Note:
The table below defines the rules, in which cases a change of a referenced end item requires a regeneration of the end item that performs the reference. The table does not define a restriction for allowed end item type references, i.e. each Cross Reference List may reference any end item type. In case there is a restriction on referenced end item types, this has to be ensured by the application e.g. CLS, FWDU, GWDU etc.

End Item type	Referenced End Item Type	Depending Aggregate of referenced EI
FWDU_SYNOPTIC_DISPLAY	EGSE_INTEGER_MEASUREMENT & EGSE_INTEGER_SW_VARIABLE	T_INTEGER_ENGINEERING_RANGE T_INTEGER_DANGER_LIMITS T_INTEGER_NOMINAL_LIMITS
	EGSE_FLOAT_MEASUREMENT & EGSE_FLOAT_SW_VARIABLE	T_FLOAT_ENG_RANGE T_FLOAT_DANGER_LIMITS T_FLOAT_NOMINAL_LIMITS T_ENGINEERING_UNITS
	EGSE_DISCRETE_MEASUREMENT & EGSE_DISCRETE_SW_VARIABLE	T_DISCRETE_MONITORING_LIST T_DISCRETE_CALIBRATION
	UNSIGNED_INTEGER_MEASUREMENT & UNSIGNED_INTEGER_SW_VARIABLE	T_UNSIGNED_INT_ENG_RANGE T_UNSIGNED_INT_DANGER_LIMITS T_UNSIGNED_INT_NOMINAL_LIMITS
	DOUBLE_FLOAT_MEASUREMENT & DOUBLE_FLOAT_SW_VARIABLE	T_DOUBLE_FLOAT_ENG_RANGE T_DOUBLE_FLOAT_DANGER_LIMITS T_DOUBLE_FLOAT_NOMINAL_LIMITS T_ENGINEERING_UNITS
	BOOLEAN_MEASUREMENT	T_BOOLEAN_MONITORING T_BOOLEAN_CALIBRATION
	BOOLEAN_SW_VARIABLE	T_BOOLEAN_MONITORING
	SWOP_COMMAND & UCL_AUTOMATED_PROCEDURE	T_FORMAL_PARAMETERS T_DEFAULT_STRING_VALUE T_DEFAULT_STATE_CODE_VALUE T_DEFAULT_INTEGER_VALUE T_DEFAULT_PATHNAME_VALUE T_DEFAULT_TIME_VALUE T_DEFAULT_REAL_VALUE T_STATE_CODE_CONSTRAINTS T_STRING_CONSTRAINTS T_INTEGER_CONSTRAINTS T_PATHNAME_CONSTRAINTS T_TIME_CONSTRAINTS T_REAL_CONSTRAINTS
	FWDU_EQUIPMENT_CONSTRAINTS	T_FWDU_TEXT
FWDU_LIST_TEXT	FWDU_EQUIPMENT_CONSTRAINTS	T_FWDU_TEXT
WDU_GROUND_SYNOPTIC_DISPLAY	EGSE_INTEGER_MEASUREMENT & EGSE_INTEGER_SW_VARIABLE	T_INTEGER_ENGINEERING_RANGE T_INTEGER_DANGER_LIMITS T_INTEGER_NOMINAL_LIMITS
	EGSE_FLOAT_MEASUREMENT & EGSE_FLOAT_SW_VARIABLE	T_FLOAT_ENG_RANGE T_FLOAT_DANGER_LIMITS T_FLOAT_NOMINAL_LIMITS
	EGSE_DISCRETE_MEASUREMENT & EGSE_DISCRETE_SW_VARIABLE	T_DISCRETE_CALIBRATION T_UNDEF_VALUES_STATE_CODE

End Item type	Referenced End Item Type	Depending Aggregate of referenced EI
CPL_SCRIPT &	UCL_SYSTEM_LIBRARY	T_UCL_SPECIFICATION_SOURCES
UCL_USER_LIBRARY &	UCL_USER_LIBRARY	T_UCL_SPECIFICATION_SOURCES
UCL_AUTOMATED_PROCEDURE &	UCL_AUTOMATED_PROCEDURE	T_UCL_AP_SOURCES
UCL_SYSTEM_LIBRARY	BOOLEAN_STIMULUS	T_FORMAL_PARAMETER_SOURCES
	DOUBLE_FLOAT_STIMULUS	T_FORMAL_PARAMETER_SOURCES
	DOUBLE_FLOAT_STIMULUS	T_ENGINEERING_UNITS
	EGSE_ANALOG_STIMULUS	T_FORMAL_PARAMETER_SOURCES
	EGSE_ANALOG_STIMULUS	T_ENGINEERING_UNITS
	EGSE_BINARY_PACKET	T_FORMAL_PARAMETER_SOURCES
	EGSE_BINARY_PACKET	T_PARA_ENGINEERING_UNITS
	EGSE_BINARY_PACKET	T_PARA_DISCRETE_CALIBRATION
	EGSE_DISCRETE_STIMULUS	T_FORMAL_PARAMETER_SOURCES
	EGSE_DISCRETE_STIMULUS	T_DISCRETE_CALIBRATION
	EGSE_PREDEFINED_TC	T_FORMAL_PARAMETER_SOURCES
	EGSE_PREDEFINED_TC	T_PARA_ENGINEERING_UNITS
	EGSE_PREDEFINED_TC	T_PARA_DISCRETE_CALIBRATION
	INTEGER_STIMULUS	T_FORMAL_PARAMETER_SOURCES
	UNSIGNED_INTEGER_STIMULUS	T_FORMAL_PARAMETER_SOURCES
	EGSE_USER_MESSAGE	T_FORMAL_PARAMETER_SOURCES
	GDU_DESCRIPTION_LIST	T_FORMAL_PARAMETER_SOURCES
	SWOP_COMMAND	T_FORMAL_PARAMETER_SOURCES
	DOUBLE_FLOAT_MEASUREMENT	T_ENGINEERING_UNITS
	EGSE_FLOAT_MEASUREMENT	T_ENGINEERING_UNITS
	EGSE_DISCRETE_MEASUREMENT	T_DISCRETE_CALIBRATION
	DOUBLE_FLOAT_SW_VARIABLE	T_ENGINEERING_UNITS
	EGSE_FLOAT_SW_VARIABLE	T_ENGINEERING_UNITS
	EGSE_FLOAT_DERIVED_VALUE	T_ENGINEERING_UNITS
	EGSE_DISCRETE_SW_VARIABLE	T_DISCRETE_CALIBRATION
	INTEGER_CONSTANT	T_INT_CONSTANTS
	REAL_CONSTANT	T_FLOAT_CONSTANTS
	STRING_CONSTANT	T_STRING_CONSTANTS
SWRU	INTEGER_CONSTANT	T_INT_CONSTANTS
	REAL_CONSTANT	T_FLOAT_CONSTANTS
	STRING_CONSTANT	T_STRING_CONSTANTS

End Item type	Referenced End Item Type	Depending Aggregate of referenced EI
EGSE_ANALOG_STIMULUS &	UCL_SYSTEM_LIBRARY	T_UCL_SPECIFICATION_SOURCES
EGSE_DISCRETE_STIMULUS &	UCL_USER_LIBRARY	T_UCL_SPECIFICATION_SOURCES
GDU_DESCRIPTION_LIST &	EGSE_DISCRETE_MEASUREMENT	T_DISCRETE_CALIBRATION
BOOLEAN_STIMULUS &	EGSE_FLOAT_MEASUREMENT	T_ENGINEERING_UNITS
DOUBLE_FLOAT_STIMULUS	EGSE_DISCRETE_SW_VARIABLE	T_DISCRETE_CALIBRATION
	EGSE_FLOAT_SW_VARIABLE	T_ENGINEERING_UNITS
	EGSE_FLOAT_DERIVED_VALUE	T_ENGINEERING_UNITS
	INTEGER_CONSTANT	T_INT_CONSTANTS
	REAL_CONSTANT	T_FLOAT_CONSTANTS
	STRING_CONSTANT	T_STRING_CONSTANTS
EGSE_PREDEFINED_TC &	UCL_SYSTEM_LIBRARY	T_UCL_SPECIFICATION_SOURCES
EGSE_BINARY_PACKET &	UCL_USER_LIBRARY	T_UCL_SPECIFICATION_SOURCES
SWOP_COMMAND	UCL_AUTOMATED_PROCEDURE	T_UCL_AP_SOURCES
	EGSE_DISCRETE_MEASUREMENT	T_DISCRETE_CALIBRATION
	EGSE_FLOAT_MEASUREMENT	T_ENGINEERING_UNITS
	EGSE_DISCRETE_SW_VARIABLE	T_DISCRETE_CALIBRATION
	EGSE_FLOAT_SW_VARIABLE	T_ENGINEERING_UNITS
	EGSE_FLOAT_DERIVED_VALUE	T_ENGINEERING_UNITS
	INTEGER_CONSTANT	T_INT_CONSTANTS
	REAL_CONSTANT	T_FLOAT_CONSTANTS
	STRING_CONSTANT	T_STRING_CONSTANTS

N-1.5 Check Minimum Number of Records

CC–11 Wrong number of records in aggregate

In the following aggregates must be exist at least “minimum number of records” for each item:

```
aggregate name                           minimum number of records
---------------------------------------- -------------------------
Analog Decalibration Point Pairs                                 2
Analog Point Pairs                                               2
Analog Point Pairs                                               2
Analog Decalibration Point Pairs                                 2
```

N-1.6 Double SID Check

CC–12 Fatal error – SID: [number] *the same SID found in end item*

• Constraint: In the configuration scope shall not exist different items with the same SID.

N-1.7 CDI Checks

The CDI checks are defined for CGS CDIs having Cross Reference Lists.

CC–13 Pathname Reference for CDI [name] *not found in actual CCU scope !*

• Constraint: A pathname reference must exist in the scope.

CC–14 Pathname Reference Change date older than End Item Change Date !

• Constraint: The change date of the referenced item must be older or equal than the generation date of the CDI.

N-1.8 CGS End Item Type related Special Checks

Raw Value Description

These checks are for all end item types having aggregate Raw Value Type.

CC–15 Invalid enumeration value

• Constraint: The Raw Value Type must correspond to the Engineering Value type of the respective measurement or stimulus as specified in following Table.

End item type	Raw Value Type	Engineering Value Type
EGSE_INTEGER_MEASUREMENT	UNSIGNED_INTEGER	INTEGER
	SIGNED_INTEGER	
EGSE_FLOAT_MEASUREMENT	UNSIGNED_INTEGER	FLOAT
	SIGNED_INTEGER	
	FLOAT	
EGSE_DISCRETE_MEASUREMENT	UNSIGNED_INTEGER	STATE_CODE
EGSE_BYTE_STREAM_MEASUREMENT	BYTESTREAM	STRING
EGSE_ANALOG_STIMULUS	FLOAT	FLOAT
EGSE_DISCRETE_STIMULUS	UNSIGNED_INTEGER	STATE_CODE
INTEGER_STIMULUS	UNSIGNED_INTEGER	INTEGER
	SIGNED_INTEGER	
BOOLEAN_STIMULUS	SIGNED_INTEGER	BOOLEAN
PULSE_STIMULUS	SIGNED_INTEGER	PULSE
BURST_PULSE_STIMULUS	UNSIGNED_INTEGER	BURST_PULSE
DOUBLE_FLOAT_STIMULUS	UNSIGNED_INTEGER	DOUBLE_FLOAT
	SIGNED_INTEGER	
UNSIGNED_INTEGER_STIMULUS	UNSIGNED_INTEGER	UNSIGNED_INTEGER

CC–16 Value out of range.

Check of aggregate Raw Value Size in Bits.

• Constraint: The Raw Value Size is dependent on the Raw Value Type of the respective measurement or stimulus as specified in following Table.

Raw Value Type	Raw Value Size in Bits
UNSIGNED_INTEGER	0 .. 31
SIGNED_INTEGER	0 .. 32
FLOAT	32

CC–17 Low value is not less than high value.

Check of the following aggregates:

Raw Value Type	Raw Value Size in Bits
UNSIGNED_INTEGER	Unsigned Integer Raw Value Range
SIGNED_INTEGER	Integer Raw Value Range
FLOAT	Float Raw Value Range

• Constraint: Low Value < High Value

Engineering Value Description

These checks are for all end item types having one of the following aggregates:
Integer Engineering Range
Unsigned Integer Engineering Range
Float Engineering Range

CC–18 Low value is not less than high value.

• Constraint: Low Value < High Value

Analog Calibration and Decalibration

These checks are for all end item types having aggregate Curve Type.

CC–19 Row*[number]* ***: Out of Raw Value Range of the corresponding end item***

Check of Analog Calibration Point Pairs.Raw Value or
Check of Analog Decalibration Point Pairs.Raw Value

• Constraint: The value of this attribute must be within the limits defined by the raw value range aggregate of the corresponding end item.

CC–20 Row*[number]* ***: Out of Engineering Value Range of the corresponding end item***

Check of Analog Calibration Point Pairs.Engineering Value or
Check of Analog Decalibration Point Pairs.Engineering Value

• Constraint: The value of this attribute must be within the limits defined by the engineering value range aggregate of the corresponding end item.

CC–21 Point pairs describe not a monotonous function

Check of Analog Calibration Point Pairs or
Check of Analog Decalibration Point Pairs

• Constraint: the sequence of N Raw Values must be monotonic over the specified raw value range
i.e. $RAW\_VALUE_i < RAW\_VALUE_{i+1}$
or $RAW\_VALUE_i > RAW\_VALUE_{i+1}$ for i=1 .. N–1

• Constraint: the sequence of N Eng. Values must be monotonic over the specified eng. value range
i.e. $ENG\_VALUE_i < ENG\_VALUE_{i+1}$
or $ENG\_VALUE_i > ENG\_VALUE_{i+1}$ for i=1 .. N–1

Discrete Calibration

These checks are for all end item types having aggregate Discrete Calibration.

*CC–22 Number of tupels of Calibration Raw Values is not less or equal 2 ** Raw Value Size in Bits*

Check of Discrete Calibration

- Constraint: The actual number of records per end item in the Discrete Calibration / Decalibration aggregate must be <= `2**Raw_Value_Size_in_Bits`, with `Raw_Value_Size_in_Bits <= 31`

CC–23 Not differs from the State Codes in Discrete Calibration aggregate

Check of Undefined Values State Code

- Constraint: For any given *discrete* end–item (measurement, stimulus, SW variable) the Undefined Values State Code, if specified, must differ from the state codes defined in the T_DISCRETE_CALIBRATION aggregate.

Byte Stream Calibration

These checks are for all end item types having aggregate Byte Stream Calibration.

CC–24 Out of range of Raw Value Size in Bytes

Check of Byte Stream Calibration.Position

- Constraint: Position must be <= Raw Value Size in Bytes of given end item

CC–25 Position + Length > Raw Value Size in Bytes + 1

Check of Byte Stream Calibration.Length

- Constraint: Position + Length <= Raw Value Size in Bytes + 1

Initial Values

These checks are for all end item types having a Initial Value aggregate.

CC–26 Out of Integer Engineering Range

Check of Initial Integer Value

- Constraint: Value must be within the range defined by the Integer Engineering Range aggregate for the particular SW Variable.

CC–27 Out of Float Engineering Range

Check of Initial Float Value

- Constraint: Value must be within the range defined by the Float Engineering Range aggregate for the particular SW Variable.

CC–28 Out of Unsigned Integer Engineering Range

Check of Initial Unsigned Integer Value

- Constraint: Value must be within the range defined by the Unsigned Integer Engineering Range aggregate for the particular SW Variable.

CC–29 Out of Double Float Engineering Range

Check of Initial Double Float Value

- Constraint: Value must be within the range defined by the Double Float Engineering Range aggregate for the particular SW Variable.

Monitoring

CC–30 End item [pathname] *musts actually have a parameter list with all parameters being optional, i.e. having default values*

Check of (xxx: Integer, Float, Unsigned Integer or Double Float):
xxx Danger Limits.High Message
xxx Danger Limits.High Action
xxx Danger Limits.Low Message
xxx Danger Limits.Low Action
xxx Danger Limits.Delta Message
xxx Danger Limits.Delta Action
xxx Nominal Limits.High Message
xxx Nominal Limits.High Action
xxx Nominal Limits.Low Message
xxx Nominal Limits.Low Action
xxx Nominal Limits.Delta Message
xxx Nominal Limits.Delta Action
Discrete Monitor List.Exception Message (Expected Value.Exception Message)
Discrete Monitor List.Exception Action (Expected Value.Exception Action)
Byte Stream Monitor List.Exception Message
Byte Stream Monitor List.Exception Action
Boolean Monitoring.Exception Message
Boolean Monitoring.Exception Action

- Constraint: If defined, the parameters of the referenced end item must all be optional with assigned default values.

CC–31 Monitoring AP [pathname] *has parameters.*

Check of

(xxx: Integer or Float):
xxx Danger Limits.High Action
xxx Danger Limits.Low Action
xxx Danger Limits.Delta Action
xxx Nominal Limits.High Action
xxx Nominal Limits.Low Action
xxx Nominal Limits.Delta Action
Discrete Monitor List.Exception Action (Expected Value.Exception Action)
Byte Stream Monitor List.Exception Action

- Constraint: If the referenced end item of type UCL_AUTOMATED_PROCEDURE than is it not allowed that the AP has parameters.

CC–32 Row [number] : Out of range of Danger Limits

Check of:
Integer Nominal Limits
Float Nominal Limits
Unsigned Integer Nominal Limits
Double Float Nominal Limits

- Constraint: Nominal High Limit <= Danger High Limit
  Nominal Low Limit => Danger Low Limit
  Nominal_Delta_Limit < Danger_Delta_Limit

CC–33 High Limit is not greater than Low Limit

Check of:
Integer Danger Limits
Float Danger Limits
Unsigned Integer Danger Nominal Limits
Double Float Danger Limits

- Constraint: Danger High Limit > Danger Low Limit

CC–34 Row [number] : High Limit is not greater than Low Limit

Check of:
Integer Nominal Limits
Float Nominal Limits
Unsigned Integer Nominal Nominal Limits
Double Float Nominal Limits

- Constraint: Nominal High Limit > Nominal Low Limit

CC–35 Expected Value [value] must be defined in Discrete Calibration.

Check of Discrete Monitor List (Expected Value)

- Constraint: If monitoring is specified in Discrete Monitor List, the expected values must be defined in Discrete Calibration.

CC–36 Out of Integer Engineering Range

Check of:
Integer Danger Limits.Danger High Limit
Integer Danger Limits.Danger Low Limit

- Constraint: Danger High Limit <= High Value in engineering range
  Danger Low Limit => Low Value in engineering range

CC–37 Out of Float Engineering Range

Check of:
Float Danger Limits.Danger High Limit
Float Danger Limits.Danger Low Limit

- Constraint: Danger High Limit <= High Value in engineering range
  Danger Low Limit => Low Value in engineering range

CC–38 Out of Unsigned Integer Engineering Range

Check of:
Unsigned Integer Danger Limits.Danger High Limit
Unsigned Integer Danger Limits.Danger Low Limit

- Constraint: Danger High Limit <= High Value in engineering range
  Danger Low Limit => Low Value in engineering range

CC–39 Out of Double Float Engineering Range

Check of:
Double Float Danger Limits.Danger High Limit
Double Float Danger Limits.Danger Low Limit

- Constraint: Danger High Limit <= High Value in engineering range
  Danger Low Limit => Low Value in engineering range

Conditions

These Checks are defined for all end item types having a condition aggregate.

CC–40 ***Operator:*** *[operator]* ***Value 1:*** *[value_1]* ***Value 2:*** *[value_2]* ***Action Type:*** *[action_type]* ***: Value 2 must be greater or equal than Value 1.***

Check of Integer Conditions.Value 2 or Float Conditions.Value 2

- Constraint: If given, Value_2_(High_value) >= Value_1_(Low_Value)

CC–41 ***Operator:*** *[operator]* ***Value 1:*** *[value_1]* ***Value 2:*** *[value_2]* ***Action Type:*** *[action_type]* ***: If Value 2 is defined then Operator must be ”in_range”***

Check of Integer Conditions.Value 2 or Float Conditions.Value 2

- Constraint: If given, Operator must be ”in_range”

CC–42 ***Operator:*** *[operator]* ***Value 1:*** *[value_1]* ***Value 2:*** *[value_2]* ***Action Type:*** *[action_type]* ***: If Operator is ”in_range” then Vaule 2 must be defined.***

Check of Integer Conditions.Value 2 or Float Conditions.Value 2

- Constraint: Must be given, if operator is ”in_range”

CC–43 ***Operator:*** *[operator]* ***Value 1:*** *[value_1]* ***Value 2:*** *[value_2]* ***Action Type:*** *[action_type]* ***:*** *[action_ei_reference]* ***is not of type UCL_AUTOMATED_PROCEDURE.***

Check of Integer Conditions.Action Enditem Reference or Float Conditions.Action Enditem Reference

- Constraint: If Action_Type is ”START_AP”, the pathname must be of type UCL_AUTOMATED_PROCEDURE.

CC–44 ***Operator:*** *[operator]* ***Value 1:*** *[value_1]* ***Value 2:*** *[value_2]* ***Action Type:*** *[action_type]* ***:*** *[action_ei_reference]* ***is of type UCL_AUTOMATED_PROCEDURE.***

Check of Integer Conditions.Action Enditem Reference or Float Conditions.Action Enditem Reference

- Constraint: If Action_Type is not ”START_AP”, the pathname must not be of type UCL_AUTOMATED_PROCEDURE.

CC–45 ***Operator:*** *[operator]* ***Value 1:*** *[value_1]* ***Value 2:*** *[value_2]* ***Action Type:*** *[action_type]* ***:*** *[action_ei_reference]* ***has no nominal limit sets defined.***

Check of Integer Conditions.Action Enditem Reference or Float Conditions.Action Enditem Reference

- Constraint: If Action_Type = 'Switch_Limit_Set', the referenced enditem must have nominal limit sets defined.

CC–46 ***Operator:*** *[operator]* ***Value 1:*** *[value_1]* ***Value 2:*** *[value_2]* ***Action Type:*** *[action_type]* ***:*** ***The given Limit Set Number*** *[limit_set_number]* ***is not defined for*** *[action_ei_reference]****.***

Check of Integer Conditions.Limit Set Number or Float Conditions.Limit Set Number

- Constraint: If Action_Type = 'Switch_Limit_Set', the given Limit Set Number must be defined for the Action_Enditem_Reference.

CC–47 ***Monitoring AP*** *[pathname]* ***has parameters.***

Check of:
Integer Conditions.Action Enditem Reference
Float Conditions.Action Enditem Reference
Discrete Conditions.Action Enditem Reference
Byte Stream Conditions.Action Enditem Reference

- Constraint: Referenced UCL_AUTOMATED_PROCEDUREs must not have parameters defined.

CC–48 ***Value*** *[value]* ***must be defined in Discrete Calibration.***

Check of Discrete Conditions.Value

- Constraint: If Discrete Conditions are specified, the values compared must be defined in Discrete Calibration.

CC–49 ***Operator:*** *[operator]* ***Value:*** *[value]* ***Action Type:*** *[action_type]* ***:*** *[action_ei_reference]* ***is not of type UCL_AUTOMATED_PROCEDURE.***

Check of Discrete Conditions.Action Enditem Reference or Byte Stream Conditions.Action Enditem Reference

- Constraint: If Action_Type is "START_AP", the pathname must be of type UCL_AUTOMATED_PROCEDURE.

CC–50 ***Operator:*** *[operator]* ***Value:*** *[value]* ***Action Type:*** *[action_type]* ***:*** *[action_ei_reference]* ***is of type UCL_AUTOMATED_PROCEDURE.***

Check of Discrete Conditions.Action Enditem Reference or Byte Stream Conditions.Action Enditem Reference

- Constraint: If Action_Type is not "START_AP", the pathname must be not of type UCL_AUTOMATED_PROCEDURE.

CC–51 ***Operator:*** *[operator]* ***Value:*** *[value]* ***Action Type:*** *[action_type]* ***:*** *[action_ei_reference]* ***has no nominal limit sets defined.***

Check of Discrete Conditions.Action Enditem Reference or Byte Stream Conditions.Action Enditem Reference

- Constraint: If Action_Type = 'Switch_Limit_Set', the referenced enditem must have nominal limit sets defined.

CC–52 Operator: *[operator]* ***Value:*** *[value]* ***Action Type:*** *[action_type]****: The given Limit Set Number*** *[limit_set_number]* ***is not defined for*** *[action_ei_reference]****.***

Check of Discrete Conditions.Limit Set Number or Byte Stream Conditions.Limit Set Number

- Constraint: If Action_Type = ’Switch_Limit_Set’, the given Limit Set Number must be defined for the Action_Enditem_Reference.

EGSE Physical Address

CC–53 Value out of range MAX_SLOT_CHANNEL

Check of EGSE Physical Address

- Constraint: The allowed range is 0 .. MAX_SLOT_CHANNEL, where MAX_SLOT_CHANNEL is an integer number dependent on the Slot Class as per table below.
  Note: Check shall be performed only when remote terminal Slot Class is not UNDEFINED or NON_STANDARD.

Slot Class	BITS_1	BITS_8	BITS_16	BITS_32	PACKET
MAX_SLOT_CHANNEL	511	63	31	15	0

EGSE_NODE

Check of end item type EGSE_NODE

CC–54 Invalid enumeration value

Check of CGS Internal Name.CGS Prefix

- Constraint: CGS Prefix must be:
  ”TES” if Node Type is TEST_NODE
  ”CSS” if Node Type is SIMULATOR
  ”HCI” if Node Type is WORKSTATION
  ”DBS” if Node Type is DATABASE_SERVER

CC–55 Combination of CGS Internal Number and Node Type is not unique

Check of CGS Internal Name.CGS Internal Number

- Constraint: CGS internal names must be unique, i.e. no two end items of type EGSE NODE may have the same CGS Prefix and Internal Number.

CC–56 Must be ’1’ for DBS node and CSS node

Check of CGS Internal Name.CGS Internal Number

- Constraint: CGS Internal Number must be 1 if CGS Prefix is ”DBS” or ”CSS”.

CC–57 Blanks or a blank name are not allowed

Check of Logical Name.Logical Name

- Constraint: Blanks or a blank name shall not be allowed

EGSE_SOFTWARE

Check of end item type EGSE_SOFTWARE

CC–58 Blanks or a blank name are not allowed

Check of Short Name.Short Name

- Constraint: The short name may not contain any space (blank) characters. It may not be the empty string.

EGSE_TEST_CONFIGURATION

Check of end item type EGSE_TEST_CONFIGURATION

CC–59 Referenced end item not describes a database server

Check of EGSE Database Node.Database Node

- Constraint: The specified pathname must refer to an existing end item of type EGSE Node with Node Type = DATABASE_SERVER.

CC–60 Referenced end item not describes a simulator

Check of Simulator Nodes.Simulator Node

- Constraint: The specified pathname must refer to an existing end item of type EGSE Node with Node Type = SIMULATOR.

CC–61 Row [entry_number] : Referenced end item [pathname] not describes a workstation

Check of EGSE Workstation Nodes.Workstation Node

- Constraint: The specified pathname must refer to an existing end item of type EGSE Node with Node Type = WORKSTATION.

CC–62 No workstation present for test configuration

Check of EGSE Workstation Nodes.Workstation Node

- Constraint: At least one workstation has to be present per test configuration.

CC–63 No workstation participates in test configuration

Check of EGSE Workstation Nodes.Is Participating

- Constraint: There must be at least one workstation participating in a given test configuration.

CC–64 Row [entry_number] : Refered end item [pathname] not describes a testnode

Check of EGSE Test Nodes.Test Node

- Constraint: The specified pathname must refer to an existing end item of type EGSE Node with Node Type = TEST_NODE.

CC–65 No testnode present for test configuration

Check of EGSE Test Nodes.Test Node

- Constraint: There must be at least one Test Node per test configuration.

CC–66 No exactly one testnode is Master Test Processor per test configuration

Check of EGSE Test Nodes.Is Master Test Processor

- Constraint: There must be one (and only one) Master Test Processor per test configuration.

CC–67 No testnode participates in test configuration

Check of EGSE Test Nodes.Is Paticipating

- Constraint: There must be at least one test node participating in a given test configuration.

CC–68 'Test configuration contains a combination of REPLAY and SIMULATION nodes

Check of EGSE Test Nodes.Execution Mode

- Constraint: A test configuration must never contain nodes executing in a combination of REPLAY and SIMULATION mode.

***CC–69 Row** [entry_number]**: End item** [pathname]* **musts actually have a parameter list with all parameters being optional, i.e. having default values**

Check of EGSE Test Nodes.Initial Automated Procedure

- Constraint: If defined, the parameters of the referenced AP must all be optional with assigned default values.

CC–70 No correspondence to aggregate EGSE Test Nodes

Check of EGSE Test Node Items.Test Node

- Constraint: This is a reference to one of the Test Nodes specified in the EGSE Test Nodes aggregate.

CC–71 Test node [pathname] EGSE Test Node Items [pathname]: Item [pathname] appears in more than one loaded item list (the item is loaded into more than one test node).

Check of EGSE Test Node Items.Loaded Item

- Constraint: If the given item appears in more than one Loaded Item List (i.e. if the items are loaded into more than one test node), then this item must not be of type:
  - EGSE_xxx_SW_VARIABLE
  - EGSE_xxx_DERIVED_VALUE
  - UNSTRUCTURED_ADU_DESCRIPTION
  - STRUCTURED_ADU_DESCRIPTION
  - CCSDS_ADU_DESCRIPTION
  - EGSE_ANALOG_STIMULUS
  - EGSE_DISCRETE_STIMULUS
  - EGSE_PREDEFINED_TC
  - EGSE_BINARY_PACKET
  - EGSE_MONITOR_LIST
  - GDU_DESCRIPTION_LIST

CC–72 No item is downloaded into testnode** [pathname]**.

Check of EGSE Test Node Items.Loaded Item

- Constraint: At least one item must be downloaded into each test node.

CC–73 End item** [pathname] **which is referenced in this GDU_DESCRIPTION_LIST is out of scope list of loaded items for test node** [pathname] **in test configuration** [pathname]**.

Check of relationships between EGSE Test Node Items, GDU items and GDU Description List items

- Constraint: The list of items loaded for one test node must contain all end items of type EGSE_xxx_STIMULUS, EGSE_PREDEFINED_TC or EGSE_BINARY_PACKET which are referenced in all end items of type GDU_DESCRIPTION_LIST in this list of items.

CC–74 SAS Reference** [pathname] **does not exist in EGSE Test Node SASs for test node** [pathname] **in test configuration** [pathname]**.

Check of relationships between EGSE Test Node Items, EGSE Test Node SASs, EGSE_xxx_STIMULUS, EGSE_PREDEFINED_TC, EGSE_BINARY_PACKET, SWOP_COMMAND, RESPONSE_PACKET and xxx_ADU_DESCRIPTION end items and items of type EGSE_SOFTWARE

- Constraint: The SASes referenced in all EGSE_xxx_STIMULUS, EGSE_PREDEFINED_TC, EGSE_BINARY_PACKET, SWOP_COMMAND, RESPONSE_PACKET and xxx_ADU_DESCRIPTION end items contained in all the specified items for one test node shall also exist in the list of ”Used SAS”.

CC–75 End item** [pathname] **is out of scope list of loaded items for test node** [pathname] **in test configuration** [pathname]**.

Check of relationships between EGSE Test Node Items, ADU_DESCRIPTION and SIMULATED_ADU_DESCRIPTION end items

- Constraint: The xxx_ADU_DESCRIPTIONs referenced in all SIMULATED_ADU_DESCRIPTION end items contained in all the specified items for one test node shall exist within the same scope.

CC–76 End item** [pathname] **is out of scope list of loaded items for test node** [pathname] **in test configuration** [pathname]**.

Check of relationships between EGSE Test Node Items, EGSE_SW_xxx_VARIABLE, ESGE_xxx_DERIVED_VALUE and EGSE_MONITOR_LIST end items

- Constraint: The list of items loaded for one test node must contain all end items of type EGSE_SW_xxx_VARIABLE or ESGE_xxx_DERIVED_VALUE which are referenced in all end items of type MONITOR_LIST and in this list of items.

CC–77 Measurement** [pathname] **is not referenced in a ADU description in the scope of loaded items for test node** [pathname] **in test configuration** [pathname]**.

Check of relationships between EGSE Test Node Items, EGSE Measurement, EGSE_MONITOR_LIST and ADU Description end items

- Constraint: The EGSE_xxx_MEASUREMENTs referenced in all MONITOR_LIST end items contained in all the specified items for one test node shall also referenced in xxx_ADU_DESCRIPTION end items contained in the same scope.

CC–78 ADU description** [pathname] **which has also a reference to measurement** [pathname] **is out of scope list of loaded items for test node** [pathname] **in test configuration** [pathname]**.

Check of relationships between EGSE Test Node Items, ADU Description and EGSE Measurement end items.

- Constraint: All xxx_ADU_DESCRIPTIONs which are references to the same EGSE_xxx_MEASUREMENT end item shall exist in the same scope of specified items for one test node.

CC–79 End item** [pathname] **is out of scope list of loaded items for test node** [pathname] **in test configuration** [pathname]**.

Check of relationships between EGSE Test Node Items, end items referenced in UCL_AUTOMATED_PROCEDURE and UCL_AUTOMATED_PROCEDURE end items

- Constraint: The UCL_USER_LIBRARY, UCL_SYSTEM_LIBRARY, EGSE_MONITOR_LIST, GDU_DESCRIPTION_LIST, EGSE_NODE, EGSE_SOFTWARE, EGSE_USER_MESSAGE, xxx_ADU_DESCRIPTION and VIRTUAL end items referenced in all UCL_AUTOMATED_PROCEDURE end items contained in all the specified items for one test node shall exist within the same scope.

CC–80 End item** [pathname] **is out of scope list of loaded items for the whole test configuration** [pathname]**.

Check of relationships between EGSE Test Node Items, end items referenced in UCL_AUTOMATED_PROCEDURE and UCL_AUTOMATED_PROCEDURE end items

- Constraint: The UCL_AUTOMATED_PROCEDURE, EGSE_xxx_SW_VARIABLE, EGSE_xxx_DERIVED_VALUE, EGSE_xxx_STIMULUS, EGSE_BINARY_PACKET and EGSE_PREDEFINED_TC end items referenced in all UCL_AUTOMATED_PROCEDURE end items contained in all the specified items for a whole test configuration shall exist within the same scope.

CC–81 Measurement** [pathname] **is not referenced in a ADU description in the scope list of loaded items for the whole test configuration** [pathname]**.

Check of relationships between EGSE Test Node Items, EGSE Measurement, EGSE SW Variable, EGSE Derived Value, GDU and ADU Description end items

- Constraint: The EGSE_xxx_MEASUREMENT end items referenced in UCL_AUTOMATED_PROCEDURE, EGSE_xxx_DERIVED_VALUE, EGSE_xxx_MEASUREMENT, EGSE_xxx_SW_VARIABLE, EGSE_xxx_STIMULUS, EGSE_PREDEFINED_TC and EGSE_BINARY_PACKET end items contained in all the specified items for a whole test configuration shall be referenced at least in one xxx_ADU_DESCRIPTION of the same scope.

CC–82 SWOP_COMMAND [pathname] : Mandatory aggregate EGSE Stimulus General Info is not defined.

Check of relationships between EGSE Test Node Items, SWOP_COMMAND and UCL_AUTOMATED_PROCEDURE end items

- Constraint: For SWOP_COMMANDS referenced in UCL_AUTOMATED_PROCEDURE end items contained in all the specified items for a whole test configuration the aggregate T_STIMULUS_GENERAL_INFO is mandatory.

CC–83 End item** [pathname] **out of scope list of loaded items for the whole test configuration** [pathname]**.

Check of relationships between EGSE Test Node Items, GDU, EGSE Derived Value, EGSE Measurement, EGSE SW Variable end items and end items which are referenced in such items

- Constraint: The end items referenced in EGSE_xxx_STIMULUS, EGSE_PREDEFINED_TC, EGSE_BINARY_PACKET, EGSE_xxx_DERIVED_VALUE, EGSE_xxx_MEASUREMENT and EGSE_xxx_SW_VARIABLE enditems contained in all the specified items for a whole test configuration shall exist within the same scope.

CC–84 Circular reference –> [pathname] –> ...

Check of relationships between EGSE Test Node Items, EGSE Derived Value end items and end items which are referenced in such items

- Constraint: No cycle shall exist within the references of EGSE_xxx_DERIVED_VALUE enditems contained in all the specified items for a whole test configuration.

CC–85 No correspondence to aggregate EGSE Test Nodes

Check of EGSE Test Node SASs.Test Node

- Constraint: This is a reference to one of the Test Nodes specified in the EGSE Test Nodes aggregate.

***CC–86 Row** [entry_numner]**: The list contains more than 20 SAS in testnode** [pathname]*

Check of EGSE Test Node SASs.Test Node

- Constraint: Up to 20 SAS may be specified per Test Node.

***CC–87 Row** [entry_number]**: Attribute is not unique in test node** [pathname]*

Check of EGSE Test Node SASs.Used SAS

- Constraint: A SAS shall only appear once in the list of SASes, i.e. every SAS is unique within one list.

EGSE_USER_MESSAGE

Check of end item type EGSE_USER_MESSAGE

CC–88 ’Number of parameters contained in the long text does not correspond to Formal Parameter List for this end item

Check of Message Text.Long Text

- Constraint: The number of parameters contained in the Long Text must correspond to the formal parameter list defined for the end item.

CC–89 The name of the** [number]**. parameter contained in the Long Text does not correspond to the Formal Parameter names defined for this end item

Check of Message Text.Long Text

- Constraint: The names (Pi) of the parameters contained in the Long Text must correspond to the formal Name of the parameters defined for the end item.

CCSDS Packet Header

CC–90 Packet Length is out of range 9 .. 4095.

Check of CCSDS Primary Header.Packet Length

- Constraint: If the CCSDS Packet has a secondary header, then the range of the Packet Length Field shall be 9 .. 4095 (since the secondary header is 10 bytes long), i.e.
  if T_CCSDS_HEADER_DESCRIPTION.**Secondary Header** = 'TRUE' then
  9 <= T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field** <= 4095

Stimulus Definition

Checks for GDUs

LOCATION :	T_GENERAL_BITSTREAM_LAYOUT. **Location**	
VALUE_SIZE:	T_INTEGER_DEFINITION.**Number of Bits**,	for integer
	32,	for float
	(*size_of* T_BINARY_DEFINITION.**Value**) * 8 ,	for strings
Packet Length Field:	T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field**	
Buffer Length or Global Length:	T_DATA_BUFF_LAYOUT_GLOB_LENGTH.**Global Length**	
N:	Number of entries (predefined values) in the data buffer	
PARAM_LOC :	T_LIST_OF_PARAMETERS.**Parameter Location**	
PARAM_SIZE:	T_LIST_OF_PARAMETERS.**Parameter Number of Bits**	

CC–91 Attribute is not unique in test node [pathname]

Check of EGSE Stimulus General Info.Private Identifier

- Constraint: The Private ID string must be unique within the list of CDUs loaded into a test node for every test node and every test configuration in a CCU.

*CC–92 Position [number]: Location + Value Size – 1 is not less or equal than (Packet Length Field + 1) * 8*

Check of EGSE_PREDEFINED_TC.General Bitstream Layout.Location

- Constraint: Each Predefined Value shall be entirely located within the bounds of the data buffer, i.e.
  LOCATION + VALUE_SIZE – 1 <= (Packet Length Field + 1) * 8

*CC–93 Location + Value Size – 1 is not less or equal than (Global Length) * 8*

Check of EGSE_BINARY_PACKET.General Bitstream Layout.Location

- Constraint: Each Predefined Value shall be entirely located within the bounds of the data buffer, i.e.
  LOCATION + VALUE_SIZE – 1 <= (Buffer Length) * 8

CC–94 Position [number_1 – number_2]: Overlapping occurs

Check of General Bitstream Layout.Location

- Constraint: Predefined Values shall not overlap one another, i.e.:
  For any Predefined Value i (i = 1 .. N–1)
  $\mathrm{LOCATION}_{(i+1)} > \mathrm{LOCATION}_{(i)} + \mathrm{VALUE\_SIZE}_{(i)} - 1$

CC–95 Position [number] value [number]: Out of range number of bits

Check of Integer Definition.Integer Value

- Constraint: The value range is limited by the number of bits defined for this entity.

CC–96 *[parameter_name]****: no entry in the Parameter Insertion List***

Check of List of Parameters.Parameter Name

- Constraint: For each formal parameter (in aggregate T_FORMAL_PARAMETERS), there must be an entry in the Parameter Insertion List (aggregate T_LIST_OF_PARAMETERS).

CC–97 *[parameter_name]****: is not compliant to the definition of the size of parameter***

Check of List of Parameters.Number of Bits

- Constraint: Depending on the SW Type of the respective parameter, the following restrictions apply:

for a Param of SW Type:	**the No. of Bits must be:**
INTEGER_TYPE	<= 32
REAL_TYPE	= 32
STRING_TYPE	<= 255*8
PATHNAME_TYPE	<= 255*8
STATE_CODE_TYPE	<= 64
TIME_TYPE	= 40

CC–98 *[parameter_name]****: Parameter Location + Parameter Size – 1 is not less or equal than (Packet Length Field + 1) * 8***

Check of EGSE_PREDEFINED_TC.List of Parameters.Location

- Constraint: Each Parameter Value shall be entirely located within the bounds of the data buffer, i.e.
  PARAM_LOC + PARAM_SIZE – 1 <= (Packet Length Field + 1) * 8

CC–99 *[parameter_name]****: Parameter Location + Parameter Size – 1 is not less or equal than (GLOBAL Length) *8***

Check of EGSE_BINARY_PACKET.List of Parameters.Location

- Constraint: Each Parameter Value shall be entirely located within the bounds of the data buffer, i.e.
  PARAM_LOC + PARAM_SIZE – 1 <= (Buffer Length) * 8

Command Verification

Check of aggregates Command Verification and Command Verification Times in GDUs

CC–100 Value is out of range 0.0 .. 86400.0.

Check of Command Verification Times.Activation Delay in Seconds

- Constraint: 0.0 <= Value <= 86400.0

CC–101 Value is out of range 0.0 .. 86400.0.

Check of Command Verification Times.Verification Timeout in Seconds

- Constraint: 0.0 <= Value <= 86400.0

CC–102 Measurement to be checked [pathname] is of type [item_type], Operator must be = or /=.

Check of Command Verification.Operator

- Constraint: If Measurement_to_be_checked is of type EGSE_DISCRETE_... , EGSE_BYTE_STREAM_... or EGSE_STRING_..., Operator must be one of = or /=.

CC–103 *[measurement_to_be_checked]**: Value** [value]**] must be convertable to a float value.***

Check of Command Verification.Value and Command Verification.High Value for range

- Constraint: If Measurement_to_be_checked is of type EGSE_FLOAT_... , strings given in Value and High_Value_for_Range must be convertable to a float value.

CC–104 *[measurement_to_be_checked]**: Value** [value]**] must be convertable to an integer value.***

Check of Command Verification.Value and Command Verification.High Value for range

- Constraint: If Measurement_to_be_checked is of type EGSE_INTEGER_..., strings given in Value and High_Value_for_Range must be convertable to an integer value.

CC–105 *[measurement_to_be_checked]**: must be limited to 8 characters (state code).***

Check of Command Verification.Value

- Constraint: If Measurement_to_be_checked is of type EGSE_DISCRETE_..., string given in Value must be limited to 8 upper–case characters (state_code).

CC–106 *[measurement_to_be_checked]**: Operator is "in_range", High Value for Range must be defined***

Check of Command Verification.High Value for range

- Constraint: If Operator is "in_range", both Value and High_Value_for_Range must ge given.

CC–107 Operator is "in_range", Measurement to be checked *[pathname]* ***must be of type EGSE_FLOAT_... or EGSE_INTEGER_....***

Check of Command Verification.Measurement to be checked

- Constraint: If Operator is "in_range", Measurement_to_be_checked must of type EGSE_FLOAT_... or EGSE_INTEGER_...

CC–108 *[measurement_to_be_checked]**: High Value for Range** [high_value]* ***must be equal or greater then Low Value for Range*** *[low_value]**.***

Check of Command Verification.High Value for range

- Constraint: If given, High_Value_for_Range >= Low_Value_for_range.

CC–109 *[measurement_to_be_checked]****: High Value for Range is defined, Operator must be ”in_range”.***

Check of Command Verification.High Value for range

• Constraint: If given, Operator must be ”in_range”.

Acquisition Data Unit (ADU)

Check of end item types:
CCSDS_ADU_DESCRIPTION
UNSTRUCTURED_ADU_DESCRIPTION
STRUCTURED_ADU_DESCRIPTION

LOCATION :	T_DATA_BUF_LAYOUT_END_ITEMS. **Location**
VALUE_SIZE:	T_RAW_VALUE_SIZE_IN_BITS.**Raw Value Size in Bits** or, for Byte Stream Measurements: T_RAW_VALUE_SIZE_IN_BYTES.**Raw Value Size in Bytes** * 8
Packet Length Field:	T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field**
N:	Number of entries (measurement values) in the data buffer
Buffer Length:	T_DATA_BUFF_LAYOUT_GLOB_LENGTH.**Global_length**

CC–110 Attribute is not unique in test node *[pathname]*

Check of ADU General Info.Private ID

• Constraint: The string has to be unique within the list of CDUs loaded into a test node for every test node and every test configuration in a CCU.

CC–111 Source is ”SECONDARY_HEADER”, the Location must be not exceed value 80.

Check of CCSDS_ADU_DESCRIPTION.Measurement End Items.Location

• Constraint: If Source is ”SECONDARY_HEADER”, the Location must not exceed the value 80.

CC–112 Source is ”HEADER”, the Location must be not exceed value 48.

Check of CCSDS_ADU_DESCRIPTION.Measurement End Items.Location

• Constraint: If Source is ”HEADER”, the Location must not exceed the value 48.

CC–113 Location *[number]* ***+*** *[pathname]****.Raw_Value_Size*** *[number]* ***– 1 is not less or equal than Global Length*** *[number]* **** 8***

Check of UNSTRUCTURED_ADU_DESCRIPTION.Measurement End Items.Location

• Constraint: Any measurement value to be contained in the ADU shall be entirely located within the bounds of the data buffer, i.e., the following condition shall be satisfied:

$$\text{LOCATION} + \text{VALUE\_SIZE} - 1 <= (\text{Buffer Length}) * 8$$

CC–114 Location *[number]* ***must be greather than previous Location*** *[number]* ***+*** *[pathname]****.Raw_Value_Size*** *[number]* ***– 1.***

Check of CCSDS_ADU_DESCRIPTION.Measurement End Items.Location and UNSTRUCTURED_ADU_DESCRIPTION.Measurement End Items.Location

• Constraint: Measurement Values shall not overlap one another, that is:
For any measurement i (i = 1 .. N–1)

$$\text{LOCATION}_{(i+1)} > \text{LOCATION}_{(i)} + \text{VALUE\_SIZE}_{(i)} - 1$$

CC–115 If the CCSDS Packet has no secondary header, then the Data Source must not be set to SECONDARY_HEADER.

Check of CCSDS_ADU_DESCRIPTION.Measurement End Items.Data Source

- Constraint: If the CCSDS Packet has no secondary header, then the Data Source must not be set to SECONDARY_HEADER.

CC–116 Location** [number] **+** [pathname]**.Raw_Value_Size** [number] **– 1 is not less or equal than (CCSDS_Primary_Header.Packet_Length** [number] **+ 1) * 8 in case Data Source is set to DATA.

or

Location** [number] **+** [pathname]**.Raw_Value_Size** [number] **– 1 is not less or equal than 6 * 8 in case Data Source is set to HEADER.

or

Location** [number] **+** [pathname]**.Raw_Value_Size** [number] **– 1 is not less or equal than 10 * 8 in case Data Source is set to SECONDARY_HEADER.

Check of CCSDS_ADU_DESCRIPTION.Measurement End Items.Location

- Constraint: Any measurement value to be contained in the ADU shall be entirely located within the bounds of the data buffer, i.e., the following condition shall be satisfied:
  LOCATION + VALUE_SIZE – 1 <= (Packet Length Field + 1) * 8
  in case the DATA_SOURCE is set to DATA
  LOCATION + VALUE_SIZE – 1 <= 6 * 8
  in case the DATA_SOURCE is set to HEADER
  LOCATION + VALUE_SIZE – 1 <= 10 * 8
  in case the DATA_SOURCE is set to SECONDARY_HEADER

CC–117 Location** [number] **and Data Source** [data_source] **:** [pathname]**.Raw_Value_Type must be UNSIGNED_INTEGER or SIGNED_INTEGER.

Check of CCSDS_ADU_DESCRIPTION.Measurement End Items.Data Source

- Constraint: If for any measurement to be contained in the ADU the DATA_SOURCE is set to HEADER or SECONDARY_HEADER, the RAW_VALUE_TYPE of the measurement must be one of the following alternatives:
  UNSIGNED_INTEGER
  SIGNED_INTEGER

CC–118 Warning: Implementation limit of the number of referenced measurements in a Structured ADU Description (68 in TES) is exceeded. The execution of this item may fail during runtime.

Check of STRUCTURED_ADU_DESCRIPTION.Measurement End Items

- Constraint: The minimal implementation limit (in TES) of the number of referenced measurements in a Structured ADU Description is 68. It exactly depends on the contents of the measurements, in particular the physical address.
If this limit is exceeded then the execution of this item may fail during runtime.

SIMULATED_ADU_DESCRIPTION

Check of end item type SIMULATED_ADU_DESCRIPTION

CC–119 Aggregate is not defined

Check of Simulated Data List

- Constraint: The Simulated Data List aggregate must be used if the Referenced ADU designates a STRUCTURED ADU Description.

CC–120 Aggregate is not defined

Check of Simulated Raw Value List

- Constraint: The Simulated Raw Value List aggregate must be used if the type of the Referenced ADU is UNSTRUCTURED or CCSDS_PACKET.

CC–121 Aggregate is not defined

Check of Simulated Data Global Length

- Constraint: The Simulated Data Global Length aggregate must be used if the Referenced ADU is UNSTRUCTURED or CCSDS_PACKET.

CC–122 Length is not less or equal than Packet Length Field + 1 of the referenced ADU Description

Check of Simulated Data Global Length.Global Length
Reference is a CCSDS_ADU_DESCRIPTON

- Constraint: Actual length of Simulated ADU must be less than or equal to the length of the referenced ADU,
i.e. if referenced ADU is a CCSDS ADU then:
Sim–ADU → T_SIMULATED_DATA_GLOBAL_LENGTH.**Global Length** <=
Ref–ADU → T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field** + 1

CC–123 Length is not less or equal than Global Length of the referenced ADU Description

Check of Simulated Data Global Length.Global Length
Reference is a UNSTRUCTURED_ADU_DESCRIPTON

- Constraint: Actual length of Simulated ADU must be less than or equal to the length of the referenced ADU,
or if referenced ADU is an unstructured ADU then:
Sim–ADU → T_SIMULATED_DATA_GLOBAL_LENGTH.**Global Length** <=
Ref–ADU → T_DATA_BUFF_LAYOUT_GLOB_LENGTH.**Global Length**

*CC–124 Location + Length – 1 is not less or equal than (Global Length) * 8*

Check of Simulated Raw Value List.Location

• Constraint: Any simulated measurement value in an Unstructured or CCSDS Simulated ADU shall be entirely located within the bounds of the data buffer, i.e., the following condition shall be satisfied:

LOCATION + VALUE_SIZE – 1 <= (Buffer Length) * 8

with

LOCATION :	T_SIMULATED_RAW_VALUE_LIST. **Location**
VALUE_SIZE:	T_SIMULATED_RAW_VALUE_LIST.**Length**
Buffer Length:	T_SIMULATED_DATA_GLOBAL_LENGTH.**Global Length**

CC–125 Row [number_1 – number2]*: Overlapping occurs*

Check of Simulated Raw Value List.Location

• Constraint: Further, measurement Values shall not overlap one another, that is:
For any measurement i (i = 1 .. N–1)

$$LOCATION_{(i+1)} > LOCATION_{(i)} + VALUE\_SIZE_{(i)} - 1$$

with

LOCATION :	T_SIMULATED_RAW_VALUE_LIST. **Location**
VALUE_SIZE:	T_SIMULATED_RAW_VALUE_LIST.**Length**
N:	Number of entries (measurement values) in the data buffer

CC–126 Row [number]*: Attribute is NULL*

Check of Simulated Raw Value List.Hexa Raw Value

• Constraint: If Raw Value in Hex is TRUE then Hexa Raw Value must be defined.

CC–127 Row** [number]**: Attribute is NULL

Check of:
Simulated Raw Value List.Integer Raw Value
Simulated Raw Value List.Unsigned Integer Raw Value
Simulated Raw Value List.Float Raw Value
Simulated Raw Value List.Bytestream Raw Value

- Constraint: The types of values specified in a Simulated ADU must be compatible with the types of the measurements at corresponding locations in the referenced ADU, i.e., the specified simulated data values (in aggregates T_SIMULATED_DATA_LIST or T_SIMULATED_RAW_VALUE_LIST) shall be:
  - either **Integer Raw Value** or **Unsigned Integer Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_INTEGER_MEASUREMENT
  - **Unsigned Integer Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_DISCRETE_MEASUREMENT
  - **Float Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_FLOAT_MEASUREMENT
  - **Byte Stream Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_BYTESTREAM_MEASUREMENT

CC–128 Row** [number]**: Is not compliant to the length

Check of Simulated Raw Value List.Hexa Raw Value

- Constraint: (1) The size of the hexadecimal string depends on the **Length** attribute above.
  It must be:
  (**Length** / 4) characters if (**Length** mod 4) = 0, or
  (**Length** / 4) + 1 characters if (**Length** mod 4) <> 0
  (2) The corresponding binary value must fit into **Length** bits, (i.e. only ’0’ and ’1’ are allowed for 1 bit, ’0’ to ’3’ for 2 bits, ’0’ to ’1F’ for 5 bits, etc).

CC–129 Row** [number]**: Is not compliant to the length

Check of Simulated Raw Value List.Integer Raw Value

- Constraint: The specified value must be within the range defined by the Length attribute, e.g. if Length =4 bits, then value range is: –8 .. 7.

CC–130 Row** [number]**: Is not compliant to the length

Check of Simulated Raw Value List.Unsigned Integer Raw Value

- Constraint: The specified value must be within the range defined by the Length attribute, e.g. if Length =3 bits, then value range is: 0 .. 7.

CC–131 Row** [number]**: Is not compliant to the length

Check of Simulated Raw Value List.Float Raw Value

- Constraint: The corresponding Length must be 32; specified value must be 32 bits long.

CC–132 The number of entries exceeds the number of measurement end items in the Measurement End Item List

Check of Simulated Data List

- Constraint: For any given Structured Simulated ADU (i.e. when type of referenced ADU = 'STRUCTURED'), the number of entries in the Simulated Data List (aggregate T_SIMULATED_DATA_LIST) must not exceed the number of measurement end items in the Measurement End Item List (aggregate T_MEASUREMENT_END_ITEM_LIST) of the referenced ADU.

CC–133 Row** [number]**: Attribute is NULL

Check of:
Simulated Data List.Integer Raw Value
Simulated Data List.Unsigned Integer Raw Value
Simulated Data List.Float Raw Value
Simulated Data List.Bytestream Raw Value

- Constraint: The types of values specified in a Simulated ADU must be compatible with the types of the measurements at corresponding locations in the referenced ADU, i.e., the specified simulated data values (in aggregates T_SIMULATED_DATA_LIST or T_SIMULATED_RAW_VALUE_LIST) shall be:
  - either **Integer Raw Value** or **Unsigned Integer Raw Value**
    if the corresponding measurement in the referenced ADU
    is of type EGSE_INTEGER_MEASUREMENT
  - **Unsigned Integer Raw Value**
    if the corresponding measurement in the referenced ADU
    is of type EGSE_DISCRETE_MEASUREMENT
  - **Float Raw Value**
    if the corresponding measurement in the referenced ADU
    is of type EGSE_FLOAT_MEASUREMENT
  - **Byte Stream Raw Value**
    if the corresponding measurement in the referenced ADU
    is of type EGSE_BYTESTREAM_MEASUREMENT

GDU_DESCRIPTION_LIST

Check of end item type GDU_DESCRIPTION_LIST

CC–134 Row** [number]**: End item** [pathname] **musts actually have a parameter list with all parameters being optional, i.e. having default values

Check of Stimuli List.End Item Reference

- Constraint: If defined, the parameters of the referenced end item must all be optional with assigned default values.

Parameter related Checks

This checks are defined for end item types having Formal Parameters.

CC–135 [parameter_name] is not conform to a Parameter Name in aggregate Formal Parameters

Check of
List of Parameters

Parameter Decalibration Curve Type
Parameter Analog Decalibration Point Pairs
Parameter Analog Calibration Coefficients
Parameter Analog Point Pairs
Parameter Discrete Decalibration
Parameter Raw Value Type
Parameter Raw Value Size in Bits
Parameter Raw Value Size in Bytes
Parameter Integer Raw Value Range
Parameter Unsigned Integer Raw Value Range
Parameter Float Raw Value Range
Parameter Integer Engineering Range
Parameter Float Engineering Range
Parameter Engineering Unit

- Constraint: This is the name of the parameter (same as in aggregate T_FORMAL_PARAMETERS).

CC–136 Parameter** [name]**: Number of Point Pairs out of range 2..20

Check of Parameter Analog Decalibration Point Pairs

- Constraint: Multi–record aggregate. Minimum: 2 , maximum: 20 point pairs per parameter, max. 255 parameter

CC–137 Parameter** [name]**:** [raw_value] **out of Raw Value Range

Check of Parameter Analog Decalibration Point Pairs.Raw Value

- Constraint: Value must be within the raw value range specified for the particular parameter.

CC–138 Parameter** [name]**: [engineering_value] out of Integer Engineering Range

Check of Parameter Analog Decalibration Point Pairs.Engineering Value

- Constraint: Value must be within the engineering value range specified for the particular parameter.

CC–139 Parameter** [name]**: [engineering_value] out of Float Engineering Range

Check of Parameter Analog Decalibration Point Pairs.Engineering Value

- Constraint: Value must be within the engineering value range specified for the particular parameter.

CC–140 Parameter** [name]**: Point pairs describe not a monotonous function

Check of Parameter Analog Decalibration Point Pairs

- Constraint: For any given parameter, the sequence of N Eng.Values must be monotonic over the specified eng. value range i.e.

  $ENG\_VALUE_i < ENG\_VALUE_{i+1}$

  or $ENG\_VALUE_i > ENG\_VALUE_{i+1}$ for i=1 .. N–1

  For any given parameter, the sequence of N Raw Values must be monotonic over the specified raw value range i.e.

  $RAW\_VALUE_i < RAW\_VALUE_{i+1}$

  or $RAW\_VALUE_i > RAW\_VALUE_{i+1}$ for i=1 .. N–1

CC–141 Parameter [name]: Number of Point Pairs out of range 2..20

Check of Parameter Analog Point Pairs

- Constraint: Multi–record aggregate. Minimum: 2 , maximum: 20 point pairs per parameter, max. 255 parameter

CC–142 Parameter [name]: [raw_value] out of Raw Value Range

Check of Parameter Analog Point Pairs.Raw Value

- Constraint: Value must be within the raw value range specified for the particular parameter.

CC–143 Parameter [name]: [engineering_value] out of Integer Engineering Range

Check of Parameter Analog Point Pairs.Engineering Value

- Constraint: Value must be within the engineering value range specified for the particular parameter.

CC–144 Parameter [name]: [engineering_value] out of Float Engineering Range

Check of Parameter Analog Point Pairs.Engineering Value

- Constraint: Value must be within the engineering value range specified for the particular parameter.

CC–145 Parameter [name]: Point pairs describe not a monotonous function

Check of Parameter Analog Point Pairs

- Constraint: For any given parameter, the sequence of N Eng.Values must be monotonic over the specified eng. value range i.e.
  $ENG\_VALUE_i < ENG\_VALUE_{i+1}$
  or $ENG\_VALUE_i > ENG\_VALUE_{i+1}$ for i=1 .. N–1
  For any given parameter, the sequence of N Raw Values must be monotonic over the specified raw value range i.e.
  $RAW\_VALUE_i < RAW\_VALUE_{i+1}$
  or $RAW\_VALUE_i > RAW\_VALUE_{i+1}$ for i=1 .. N–1

CC–146 Invalid enumeration value

Check of Parameter Raw Value Type.Raw Value Type

- Constraint: The allowed Raw Value Type depends on the **SW type** of the particular parameter as indicated in the following table:

SW Type of Formal Parameter	Allowed Raw Value Type
INTEGER_TYPE	SIGNED_INTEGER
	UNSIGNED_INTEGER
UNSIGNED_INTEGER_TYPE	SIGNED_INTEGER
	UNSIGNED_INTEGER
REAL_TYPE	SIGNED_INTEGER
	UNSIGNED_INTEGER
	FLOAT
STATE_CODE_TYPE	UNSIGNED_INTEGER
STRING_TYPE	BYTE_STREAM

CC–147 Parameter [name]: Value out of range.

Check of Parameter Raw Value Size in Bits.Raw Value Size in Bits

- Constraint: The allowed Raw Value Size depends on the **Raw Value Type** as indicated in the following table:

Raw Value Type	Raw Value Size in Bits
SIGNED_INTEGER	1 .. 32
UNSIGNED_INTEGER	1 .. 31
FLOAT	32
BYTE_STREAM	N/A

CC–148 Parameter [name]: High Value is not greater than Low Value

Check of:
Parameter Integer Raw Value Range.High Value
Parameter Unsigned Integer Raw Value Range.High Value
Parameter Float Raw Value Range.High Value

- Constraint: Low Value < High Value

CC–149 Parameter [name]: High Value is not greater than Low Value

Check of:
Parameter Integer Engineering Range.High Value
Parameter Float Engineering Range.High Value

- Constraint: Low Value < High Value

CC–150 Parameter [name]: Value must be equal or greater than 0 for a unsigned integer parameter.

Check of Parameter Integer Engineering Range.Low Value

- Constraint: If the SW Type of Formal Parameter is UNSIGNED_INTEGER_TYPE then the Low Value must be equal or greater than 0.

CC–151 [state_code] is not unique in parameter [name]

Check of Parameter Discretet Decalibration.State Code

- Constraint: must be unique for one given stimulus

*CC–152 Parameter [name]: Number of calibration raw values is not less or equal than 2 ** raw value size in bits*

Check of Parameter Discretet Decalibration

- Constraint: must be <= 2**Parameter Raw Value Size in Bits

CC–153 Parameter [name]: Too many calibration raw value/state code pairs (>32)

Check of Parameter Discretet Decalibration

- Constraint: Maximum: 32 raw value/state code pairs per parameter, max 255 parameter

CC–154 too many parameters (> 255)

Check of Formal Parameter for end item types:
EGSE_PREDEFINED_TC
EGSE_BINARY_PACKET
SWOP_COMMAND

• Constraint: Up to 255 formal parameters are allowed.

APID

Check of end item type APID

CC–155 Identification [ID]: Combination of Source CCSDS End Point and Destination CCSDS End Point is not unique in configuration scope

Check of Apid Table.Source CCSDS End Point and Apid Table.Destination CCSDS End Point

• Constraint: The combination of the two attributes Source CCSDS End Point and Destination CCSDS End Point must be unique within the configuration scope.

CC–156 Application ID [ID]: Combination of Application ID and Type is not unique in configuration scope

Check of Apid Table.Application ID and Apid Table.Type

• Constraint: The combination of the two attributes Application ID and Type must be unique within the configuration scope.

RESPONSE_PACKET

Check of end item type RESPONSE_PACKET

CC–157 Attribute is not unique in test node [name]

Check of Response Packet.Private ID

• Constraint: The Private ID string must be unique within the list of CDUs loaded into a test node for every test node and every test configuration in a CCU.

SWOP_COMMAND

Check of end item type SWOP_COMMAND

CC–158 Value must be TRUE

Check of CCSDS Second Header.Checksum Indicator

• Constraint: Checksum Indicator in aggregate CCSDS Second Header must be TRUE.

CC–159 Value must be ”System_Command” or ”Essential_Command” or ”Payload_Command”.

Check of CCSDS Second Header.Packet Type

• Constraint: Packet Type in CCSDS Second Header must be ”System_Command” or ”Essential_Command” or ”Payload_Command”.

CC–160 Value must be NO_TIME_FIELD or TIME_OF_PACKET_GENERATION.

Check of CCSDS Second Header.Time ID

- Constraint: Time ID in CCSDS Second Header must be NO_TIME_FIELD or TIME_OF_PACKET_GENERATION.

SWRU

Check of end item type SWRU

CC–161 Initialization and Exchange AP shall not refer to the same AP.

Check of General.Initialization AP

- Constraint: Initialization and Exchange APs shall not refer to the same AP.

N-2 List of Single Enditem Checks (Check MDB Item)

The Check_MDB_Item program is implemented for the following enditem types:

MEAS	EGSE_xxx_MEASUREMENT EGSE_xxx_SW_VARIABLE EGSE_xxx_DERIVED_VALUE
ADU	xxx_ADU_DESCRIPTION
GDU	EGSE_PREDEFINED_TC EGSE_BINARY_PACKET EGSE_ANALOG_STIMULUS EGSE_DISCRETE_STIMULUS
SYNOPT	WDU_GROUND_SYNOPTIC_DISPLAY

The checks as listed in the following table are defined:

ID	ENDI-TEM CLASS	CHECK CLASS	MSG TYPE	MESSAGE / Description
C1	MEAS	RANGE	ERROR	Danger Limits not in Engineering Range
C2	MEAS	RANGE	ERROR	Nominal Limit not in Engineering Range
C5	MEAS	RANGE	ERROR	Danger High Limit < Danger Low Limit
C10	MEAS	RANGE	ERROR	Nominal High Limit < Nominal Low Limit
C12	MEAS	RANGE	ERROR	Danger High Limit < Nominal High Limit
C13	MEAS	RANGE	ERROR	Danger Low Limit > Nominal Low Limit
C14	MEAS	RANGE	ERROR	Expected Value not defined as Statecode in Calibration
C15	MEAS	RANGE	WARN	Raw value range does not include 0 – dangerous for calibration of uninitialized values
C20	MEAS	RANGE	ERROR	Low Value of Engineering Range >= High Value of Engineering Range
C21	MEAS	RANGE	ERROR	Low Value of Engineering Range > Initial Value > High Value of Engineering Range
C22	MEAS	RANGE	ERROR	Alarm Counter range 1..10
C23	MEAS	RANGE	ERROR	Condition: In_range: Value_2(High_Value) is lower than Value_1 (Low_Value)
C1	MEAS	CALIB	ERROR	Wrong calibration definitions
C2	MEAS	CALIB	ERROR	Raw Value in discrete calibration > 2** raw_value_size_in_bits
C7	MEAS	CALIB	ERROR	OTHER must not be used as a normal calibration code
C8*	MEAS	CALIB	ERROR	Condition: Referenced Statecode not in calibration state code list
C1	MEAS	MANDAT	WARN	Initial Value is 0 (in DB: undefined or 0) – Might be unwanted

C2	MEAS	MANDAT	ERROR	Engineering Range: Optional.
				  Default is min–value .. max_value
				Raw Value Range: Optional.
				  Default is min–value .. max_value
				Calibration
				Raw Value Type / Raw Value Size

C3*	MEAS	MANDAT	ERROR	If calibration is given, raw value definition is mandatory for loading sw – even if not used in GWDU (applies to discrete sw variables/derived values only)
C3	MEAS	REFER	ERROR	Undefined/Wrong References in Derived Value Expressions
C4	MEAS	REFER	ERROR	Conditions: Wrong/Undefined References in Conditions Referenced AP must not have parameters

ID	ENDI-TEM CLASS	CHECK CLASS	MSG TYPE	MESSAGE / Description
C1	MEAS	REFER	ERROR	Reference for Action/Message invalid –Reference in Monitoring Action Descriptions: AP,GDU,GDU_List – Reference in Monitoring Action Descriptions: AP without Parameter – Reference in Monitoring Message Descriptions: EGSE_User_Message
C5	MEAS	REFER	ERROR	Conditions: Referenced Enditem Type not compatible with Comparator
C1	MEAS	MISC	ERROR	Danger Limits cannot be defined without nominal limits
C2*	MEAS	MISC	ERROR	Engineering Unit is not valid
C1	GDU	RANGE	ERROR	Number of Formal Parameters must be <= 255
C2	GDU	RANGE	ERROR	For each Formal Parameter there must be an entry in List_of_Parameters
C3	GDU	RANGE	ERROR	For each Parameter the raw value size must conform to Number of Bits in List_of_Parameters
C4	GDU	RANGE	ERROR	Parameter Values: Location + Size –1 <= Packet length*8
C5	GDU	RANGE	ERROR	Predefined Values: location + number_of_bits – 1 <= (number of bytes in packet)*8
C6	GDU	RANGE	ERROR	Length in CCSDS header: 9 <= length <= 4095 resp. 1 <= length <= 4095 (if no 2nd header)
C7	GDU	RANGE	ERROR	Maximum Length acc. to Parametertype”)); 1 <= length <= 31 for statecodes 8 <= length <= 4096 for strings 1 <= length <= 32 for integer length = 32 for pathnames length = 32 for floats (real) length = 40 for time
C8	GDU	RANGE	ERROR	GDU Verification: Time value is negative –> Verification will be ignored
C9	GDU	RANGE	ERROR	GDU Verification: In_Range, but high value < low value —> Verification will be ignored
C1	GDU	MANDAT	ERROR	SAS Reference in General Info
C2	GDU	MANDAT	ERROR	CCSDS Primary Header Fields
C3	GDU	MANDAT	WARN	CCSDS Secondary Header: Packet ID not defined: is set to GDU SID: Might be unwanted
C4	GDU	MANDAT	ERROR	Analog/Discrete Stimulus must have exactly 1 Parameter: Is parameter definition compiled?
C5	GDU	MANDAT	ERROR	Bitstream Layout Attributes are all mandatory

ID	ENDI-TEM CLASS	CHECK CLASS	MSG TYPE	MESSAGE / Description
C6	GDU	MANDAT	ERROR	Definition Attributes are all mandator for INTEGER_DEFINITION, FLOAT_DEFINITION, BINARY_DEF.
C1	GDU	CALIB	ERROR	Analog Decalibration: Curve Type is POINT_PAIRS, but no point pairs given
C2	GDU	CALIB	ERROR	Decalibration defined, but no raw value / Raw value type does not conform to decalibration
C3	GDU	CALIB	ERROR	Engineering Range for Parameter not valid / not defined
C4	GDU	CALIB	ERROR	Raw Value Range for Parameter not valid / not defined
C5*	GDU	CALIB	ERROR	Condition: Referenced Statecode not in calibration state code list
C1	GDU	REF	ERROR	GDU Verification: Reference to wrong enditem type
C2	GDU	REF	ERROR	GDU Verification: Referenced Enditem Type not compatible with Comparator
C1	GDU	MISC	ERROR	Type of Parameters must be of mode IN
C2	GDU	MISC	ERROR	Type of Parameters must be one of STRING,STATE_CODE,INTEGER,UNSIGNED_INTEGER,REAL,PATHNAME,TIME
C3	GDU	MISC	ERROR	Parameter is optional, but has no default value
C1	ADU	RANGE	ERROR	Length in CCSDS header: 9 <= length <= 4095 resp. 1 <= length <= 4095 (if no 2nd header
C2	ADU	RANGE	ERROR	Global Length / Bit Locations in Unstructured ADUs : 1<= location <= global length
C1	ADU	MANDAT	ERROR	SAS Reference in General Info
C2	ADU	MANDAT	ERROR	CCSDS Primary Header Fields
C3	ADU	MANDAT	ERROR	CCSDS Secondary Header: Packet ID not defined: is set to ADU SID: Might be unwanted
C1	ADU	REF	ERROR	Referenced measurement not found in selected scope
C2	ADU	REF	ERROR	Referenced item is not of type EGSE_xxx_MEASUREMENT
C3	ADU	REF	WARN	ADU Descriptor does not contain any reference to a measurement
C1	ADU	MISC	WARN	Overlapping of Bit Positions for measurements in ADU

ID	ENDI-TEM CLASS	CHECK CLASS	MSG TYPE	MESSAGE / Description
C1	SYNOPT	REF	ERROR	Referenced variable (measurement) not found in selected scope
C2	SYNOPT	REF	ERROR	Referenced synoptic (for Display Action) not found in selected scope

* => check is currently not implemented